Contents

Issue 1, June 1994 (1994-06-01) 5
Welcome to \LaTeX News 5
\LaTeX \textsc{2ε}—the new \LaTeX release 5
Why a new \LaTeX? 5
Processing documents with \LaTeX \textsc{2ε} 5
New packages 5
Further information 5

Issue 2, December 1994 (1994-12-01) 6
Welcome to \LaTeX News 2 6
December 1994 release of \LaTeX 6
Accented input 6
AMS-\LaTeX 6
\LaTeX on the internet 6
Further information 6

Issue 3, June 1995 (1995-06-01) 7
Welcome to \LaTeX News 3 7
June 1995 release of \LaTeX 7
Additional input encodings 7
\LaTeX getting smaller 7
Distribution and modification 7
AMS-\LaTeX full release 7
PostScript fonts 7
Further information 7

Issue 4, December 1995 (1995-12-01) 8
Welcome to \LaTeX News 4 8
\LaTeX getting smaller 8
New ‘concurrent’ docstrip 8
New \textsc{T1} encoded fonts 8
More robust commands 8
New Interface to building ‘extension’ classes 8
More Input Encodings 8
Further information 8

Issue 5, June 1996 (1996-06-01) 9
Welcome to \LaTeX News 5 9
Extra possibilities for section headings 9
The ‘openany’ option in the ‘book’ class 9
More font (output) encodings 9
More input encodings supported 9
Fixes and improvements 9
Changes to the ‘tools’ packages 9
New copy of the \LaTeX bug database 9

Issue 6, December 1996 (1996-12-01) 11
Welcome to \LaTeX News 6 11
Mono-case file names 11
Another input encoding 11
Better user-defined math display environments 11
Docstrip improvements 11
AMS \LaTeX update 11
Graphics package update 11
EC Fonts released 11

Issue 7, June 1997 (1997-06-01) 12
T1 encoded Computer Modern fonts 12
T1 encoded Concrete fonts 12
Further input encodings 12
Normalising spacing after punctuation 12
Accessing Bold Math Symbols 12
Policy on standard classes 12
New addresses for TUG 12

Issue 8, December 1997 (1997-12-01) 13
New supported font encodings 13
New input encodings 13
Tools 13
Graphics 13
\LaTeXX3 experimental programming conventions 13

Issue 9, June 1998 (1998-06-01) 14
New math font encodings 14
A new math accent 14
Extended \texttt{\textbackslash DeclareMathDelimiter} 14
Tools distribution 14
Support for Cyrillic encodings 14
Default docstrip header 14

Issue 10, December 1998 (1998-12-01) 15
Five years of \LaTeX \textsc{2ε} 15
Restructuring the \LaTeX distribution 15
\LaTeX Project on the Internet 15
Restructuring the \LaTeX package licenses 15
Support for Cyrillic encodings 15
Tools distribution 15

Issue 11, June 1999 (1999-06-01) 16
Back in sync 16
Yearly release cycles 16
LPPL update 16
The future of S\textsc{li}\LaTeX 16
Fontenc package peculiarities 16
New math font encodings 16
Tools distribution 16
Coming soon 16
<table>
<thead>
<tr>
<th>Issue 12, December 1999 (1999-12-01)</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPPL update</td>
<td>17</td>
</tr>
<tr>
<td>fixltx2e package</td>
<td>17</td>
</tr>
<tr>
<td>Outcome of TUG '99 (Vancouver)</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 13, June 2000 (2000-06-01)</th>
<th>18</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yearly release cycle</td>
<td>18</td>
</tr>
<tr>
<td>PSNFSS: Quote of the Month</td>
<td>18</td>
</tr>
<tr>
<td>New AMS-\LaTeX</td>
<td>18</td>
</tr>
<tr>
<td>New input encoding \texttt{latin4}</td>
<td>18</td>
</tr>
<tr>
<td>New experimental code</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 14, June 2001 (2001-06-01)</th>
<th>19</th>
</tr>
</thead>
<tbody>
<tr>
<td>Future releases</td>
<td>19</td>
</tr>
<tr>
<td>New release of Babel (required)</td>
<td>19</td>
</tr>
<tr>
<td>New input encoding \texttt{latin9}</td>
<td>19</td>
</tr>
<tr>
<td>New tools</td>
<td>19</td>
</tr>
<tr>
<td>New experimental code</td>
<td>19</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 15, December 2003 (2003-12-01)</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anniversary release</td>
<td>20</td>
</tr>
<tr>
<td>LPPL – new version</td>
<td>20</td>
</tr>
<tr>
<td>Small updates to varioref</td>
<td>20</td>
</tr>
<tr>
<td>New and more robust commands</td>
<td>20</td>
</tr>
<tr>
<td>Fixing font sizes</td>
<td>20</td>
</tr>
<tr>
<td>Font encodings</td>
<td>20</td>
</tr>
<tr>
<td>Displaying font tables</td>
<td>20</td>
</tr>
<tr>
<td>New input encodings</td>
<td>20</td>
</tr>
<tr>
<td>Unicode input</td>
<td>20</td>
</tr>
<tr>
<td>And finally ... pict2e</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 16, December 2003 (2003-12-01)</th>
<th>21</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anniversary news</td>
<td>21</td>
</tr>
<tr>
<td>TLC2: The \LaTeX\ Companion – 2nd edition!</td>
<td>21</td>
</tr>
<tr>
<td>Future maintenance</td>
<td>21</td>
</tr>
<tr>
<td>LPPL certification</td>
<td>21</td>
</tr>
<tr>
<td>Use of \texttt{\v TeX}/\pdftex</td>
<td>21</td>
</tr>
<tr>
<td>End of ‘autoload’ support</td>
<td>21</td>
</tr>
<tr>
<td>New models, new code</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 17, December 2005 (2005-12-01)</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project licence news</td>
<td>22</td>
</tr>
<tr>
<td>New guide on font encodings</td>
<td>22</td>
</tr>
<tr>
<td>Robust commands in math</td>
<td>22</td>
</tr>
<tr>
<td>Updates of required packages</td>
<td>22</td>
</tr>
<tr>
<td>Work on \LaTeX\ fixes</td>
<td>22</td>
</tr>
<tr>
<td>The graphics bundle</td>
<td>23</td>
</tr>
<tr>
<td>Future development</td>
<td>23</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 18, December 2007 (2007-12-01)</th>
<th>24</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Issue 19, September 2009 (2009-09-24)</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>New \LaTeX\ release</td>
<td>25</td>
</tr>
<tr>
<td>New code repository</td>
<td>25</td>
</tr>
<tr>
<td>Babel</td>
<td>25</td>
</tr>
<tr>
<td>The future</td>
<td>25</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 20, June 2011 (2011-06-27)</th>
<th>26</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduled \LaTeX\ bug-fix release</td>
<td>26</td>
</tr>
<tr>
<td>Continued development</td>
<td>26</td>
</tr>
<tr>
<td>Release notes</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 21, May 2014 (2014-05-01)</th>
<th>28</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scheduled \LaTeX\ bug-fix release</td>
<td>28</td>
</tr>
<tr>
<td>Standard \LaTeX\ (\LaTeX\ 2ε) and expl3</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 22, January 2015 (2015-01-01)</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>New \LaTeX\ 2ε bug-fix policy</td>
<td>30</td>
</tr>
<tr>
<td>Updates to the kernel</td>
<td>31</td>
</tr>
<tr>
<td>\texttt{\LaTeX}build</td>
<td>31</td>
</tr>
<tr>
<td>Hyperlinked documentation and TDS zip files</td>
<td>31</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 23, October 2015 (2015-10-01)</th>
<th>32</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced support for \LuaTeX\</td>
<td>32</td>
</tr>
<tr>
<td>More Floats and Inserts</td>
<td>33</td>
</tr>
<tr>
<td>Updated Unicode data</td>
<td>33</td>
</tr>
<tr>
<td>Support for Comma Accent</td>
<td>33</td>
</tr>
<tr>
<td>Extended \texttt{inputenc}</td>
<td>33</td>
</tr>
<tr>
<td>Pre-release Releases</td>
<td>33</td>
</tr>
<tr>
<td>Updates in tools</td>
<td>33</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 24, February 2016 (2016-02-01)</th>
<th>34</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LuaTeX\ support</td>
<td>34</td>
</tr>
<tr>
<td>Unicode data</td>
<td>34</td>
</tr>
<tr>
<td>More support for east European accents</td>
<td>35</td>
</tr>
<tr>
<td>Changes in Graphics</td>
<td>35</td>
</tr>
<tr>
<td>Changes in Tools</td>
<td>35</td>
</tr>
<tr>
<td>Improving support for Unicode engines</td>
<td>35</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 25, March 2016 (2016-03-01)</th>
<th>36</th>
</tr>
</thead>
<tbody>
<tr>
<td>\LaTeX\</td>
<td>36</td>
</tr>
<tr>
<td>Documentation checksums</td>
<td>36</td>
</tr>
<tr>
<td>Updates to \texttt{inputenc}</td>
<td>36</td>
</tr>
<tr>
<td>Updates in Tools</td>
<td>36</td>
</tr>
<tr>
<td>amsmath</td>
<td>36</td>
</tr>
<tr>
<td>Related updates</td>
<td>36</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Issue 26, January 2017 (2017-01-01)</th>
<th>37</th>
</tr>
</thead>
<tbody>
<tr>
<td>\v TeX</td>
<td>37</td>
</tr>
<tr>
<td>Default encodings in X\v TeX and \LaTeX\</td>
<td>37</td>
</tr>
<tr>
<td>\texttt{\showhyphens} in X\v TeX</td>
<td>38</td>
</tr>
<tr>
<td>The fixltx2e package</td>
<td>38</td>
</tr>
<tr>
<td>The \latex\2e package</td>
<td>38</td>
</tr>
<tr>
<td>Updates to amsmath</td>
<td>38</td>
</tr>
<tr>
<td>Updates to tools</td>
<td>38</td>
</tr>
</tbody>
</table>
An addendum to the release changes in 2015: page breaks and vertical spacing .......... 38

**Issue 27, April 2017 (2017-04-15)**  39
ISO 8601 Date format .................................. 39
Further TU encoding improvements .......... 39
Disabling hyphenation ................................ 39
Discretionary hyphenation ......................... 39
Default document language ....................... 39
Line spacing in parboxes ......................... 39

**Issue 28, April 2018 (2018-04-01)**  40
A new home for \texttt{\LaTeX} errors sources .......... 40
Bug reports for core \texttt{\LaTeX} errors .......... 40
UTF-8: the new default input encoding .......... 40
A general rollback concept ....................... 41
Integration of \texttt{remreset} and \texttt{chngcnt} packages .......... 42
Testing for undefined commands ............... 42
Changes to packages in the \texttt{tools} category .......... 42
Changes to packages in the \texttt{amsmath} category .......... 42

**Issue 29, December 2018 (2018-12-01)**  43
Introduction ........................................ 43
Bug reports for core \texttt{\LaTeX} errors and packages .......... 43
Changes to the \texttt{\LaTeX} kernel ................. 43
Changes to packages in the \texttt{tools} category .......... 45
Changes to packages in the \texttt{amsmath} category .......... 45
Website updates .................................... 45

**Issue 30, October 2019 (2019-10-01)**  47
\texttt{\LaTeX}-dev formats now available .......... 47
Improving Unicode handling in \texttt{pdflatex} .......... 48
Improving file name handling in \texttt{pdflatex} .......... 48
Improving the \texttt{filecontents} environment .......... 48
Making more user commands robust ............. 48
Other changes to the \texttt{\LaTeX} kernel .......... 49
Changes to packages in the \texttt{tools} category .......... 50
Changes to packages in the \texttt{amsmath} category .......... 50
Documentation updates ............................ 50

**Issue 31, February 2020 (2020-02-02)**  51
Experiences with the \texttt{\LaTeX}-dev formats .......... 51
Concerning this release \ldots (Lua\texttt{\LaTeX} engine) .......... 51
Improved load-times for \texttt{expl3} .............. 51
Improvements to \texttt{\LaTeX} font selection: NFSS .......... 52
Other changes to the \texttt{\LaTeX} kernel .......... 54
Changes to packages in the \texttt{graphics} category .......... 55
Changes to packages in the \texttt{tools} category .......... 55
\texttt{\LaTeX} requirements on engine primitives .......... 55

**Issue 32, October 2020 (2020-10-01)**  56
Introduction ........................................ 56
Providing \texttt{xparse} in the format ................ 56
A hook management system for \texttt{\LaTeX} .......... 57
Other changes to the \texttt{\LaTeX} kernel .......... 57
Changes to packages in the \texttt{graphics} category .......... 60
Changes to packages in the \texttt{tools} category .......... 61
Changes to packages in the \texttt{amsmath} category .......... 61
Changes to the \texttt{babel} package ............... 62

**Issue 33, June 2021 (2021-06-01)**  63
Introduction ........................................ 63
Extending the hook concept to paragraphs .......... 63
Extending the hook concept to commands .......... 64
Other hook business ................................ 64
Improved handling of file names ............... 64
Updates to the font selection scheme .......... 65
Glyphs, characters & encodings .................... 65
New or improved commands ....................... 66
Code improvements ................................ 67
Changes to packages in the \texttt{graphics} category .......... 68
Changes to packages in the \texttt{tools} category .......... 68
Changes to packages in the \texttt{amsmath} category .......... 68

**Issue 34, November 2021 (2021-11-15)**  69
Introduction ........................................ 69
Hook business ..................................... 69
New or improved commands ....................... 71
Code improvements ................................ 72
Bug fixes .......................................... 73
Changes to packages in the \texttt{amsmath} category .......... 73
Changes to packages in the \texttt{graphics} category .......... 74
Changes to packages in the \texttt{tools} category .......... 74

**Issue 35, June 2022 (2022-06-01)**  75
Introduction ........................................ 75
Document metadata interface ..................... 75
The \texttt{latex-lab} bundle ....................... 76
A new mark mechanism for \texttt{\LaTeX} .......... 76
A key/value bundle to option handling .......... 77
New or improved commands ....................... 77
Code improvements ................................ 78
Bug fixes .......................................... 80
Changes to packages in the \texttt{amsmath} category .......... 81
Changes to packages in the \texttt{graphics} category .......... 81
Changes to packages in the \texttt{tools} category .......... 81

**Issue 36, November 2022 (2022-11-01)**  83
Introduction ........................................ 83
Auto-detecting key/value arguments ............. 83
A note for font package developers ............. 83
New or improved commands ....................... 84
Code improvements ................................ 84
Bug fixes .......................................... 85
Changes to packages in the \texttt{graphics} category .......... 85
Changes to packages in the tools category . . . 85

**Issue 37, June 2023 (2023-06-01)** 87
New functionality offered as part of the “\LaTeX
Tagged PDF” project . . . . . . . . . . . . . . 87
New or improved commands . . . . . . . . . . . 88
Code improvements . . . . . . . . . . . . . . . 89
Bug fixes . . . . . . . . . . . . . . . . . . . . . 90
Documentation improvements . . . . . . . . . . 91
Changes to packages in the tools category . . . 92

**Issue 38, November 2023 (2023-11-01)** 93
News from the “\LaTeX Tagged PDF” project . 93
Hooks, sockets and plugs . . . . . . . . . . . . . 93
Document properties and cross-referencing . . 94
New or improved commands . . . . . . . . . . . 95
Code improvements . . . . . . . . . . . . . . . 95
Removed kernel commands . . . . . . . . . . . 96
Changes to packages in the tools category . . . 96

**Issue 39, June 2024 (2024-06-01)** 97
Introduction . . . . . . . . . . . . . . . . . . . 97
News from the “\LaTeX Tagged PDF” project . 97
Enhancements to the new mark mechanism . . 98
Providing xtemplate in the format . . . . . . . 98
New or improved commands . . . . . . . . . . 99
Code improvements . . . . . . . . . . . . . . . 99
Documentation improvements . . . . . . . . . 101
Bug fixes . . . . . . . . . . . . . . . . . . . . . 101
Changes to packages in the amsmath category . 102
Changes to packages in the tools category . . 102
Changes to files in the cyrillic category . . . 103
Welcome to \LaTeX News

An issue of \LaTeX News will accompany every future release of \LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other \LaTeX news.

\LaTeX 2ε—the new \LaTeX release

The most important news is the release of \LaTeX 2ε, the new version of the \LaTeX software. This version has better support for fonts, graphics and colour, and will be actively maintained by the \LaTeX Project team. Upgrades will be issued every six months, in June and December.

Why a new \LaTeX?

Over the years many extensions have been developed for \LaTeX. This is, of course, a sure sign of its continuing popularity but it has had one unfortunate result: incompatible \LaTeX formats came into use at different sites. Thus, to process documents from various places, a site maintainer was forced to keep \LaTeX (with and without NFSS), \SLATEX, \AMSTEX, and so on. In addition, when looking at a source file it was not always clear for which format the document was written.

To put an end to this unsatisfactory situation a new release of \LaTeX was produced. It brings all such extensions back under a single format and thus prevents the proliferation of mutually incompatible dialects of \LaTeX 2.09. The new release was available for several months as a test version, and the final release of 1 June officially replaces the old version.

Processing documents with \LaTeX 2ε

Documents written for \LaTeX 2.09 will still be read by \LaTeX 2ε. Any such document is run in \LaTeX 2.09 compatibility mode.

Unfortunately, compatibility mode comes with a price: it can run up to 50% slower than \LaTeX 2.09 did. If you want to run your document in the faster native mode, you should try replacing the line:

\begin{verbatim}
documentstyle[options,packages]{class}
\end{verbatim}

with:

\begin{verbatim}
documentclass[options]{class}
\usepackage{latexsym,packages}
\end{verbatim}

Unfortunately, this will not always work, because some \LaTeX 2.09 packages will only work in \LaTeX 2ε compatibility mode. You should find out if there is a \LaTeX 2ε version of the package available.

\LaTeX 2ε native mode also gives access to the new features of \LaTeX 2ε, described in \LaTeX 2ε for authors.

New packages

\LaTeX 2ε has much better support for graphics, colour, fonts, and multi-lingual typesetting. The following software should be available from the distributor who brought you \LaTeX 2ε:

- babel, for typesetting in many languages.
- color, for colour support.
- graphics, for including images.
- mfnfss, for using bitmap fonts.
- psnfss, for using Type 1 fonts.
- tools, other packages by the \LaTeX Project team.

The packages come with full documentation, and are also described in \LaTeX: A Document Processing System or The \LaTeX Companion.

Further information

More information about \LaTeX 2ε is to be found in:

- The \LaTeX Companion, Goossens, Mittelbach and Samarim, Addison Wesley, 1994.

The \LaTeX distribution comes with documentation on the new features of \LaTeX:

- \LaTeX 2ε for authors, describes the new features of \LaTeX documents, in the file \texttt{usrguide.tex}.
- \LaTeX 2ε for class and package writers, describes the new features of \LaTeX classes and packages, in the file \texttt{clsguide.tex}.
- \LaTeX 2ε font selection, describes the new features of \LaTeX fonts for class and package writers, in the file \texttt{fntguide.tex}.

For more information on \TeX and \LaTeX, get in touch with your local \TeX Users Group, or the international \TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.
Welcome to \LaTeX News 2

An issue of \LaTeX News will accompany every future release of \LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other \LaTeX news.

December 1994 release of \LaTeX

December 1994 sees the second release of \LaTeX2ε. We are on schedule to deliver a release of \LaTeX every six months, in December and June.

This release has seen quite a lot of activity, which is not too surprising as it’s only been a year since the first test release of \LaTeX2ε. We don’t expect so much activity in the next six months.

Many of the changes are minor improvements and bug-fixes—see \LaTeX2ε for authors (usrguide.tex), \LaTeX2ε font selection (fntguide.tex) and our change log (changes.txt) for more details.

However, there are two important new packages available for \LaTeX: \inputenc and AMS-\LaTeX.

Accented input

One of the problems with writing non-English documents in \LaTeX is the accent commands. Reading documents containing text like na"ive is frustrating, especially if your keyboard allows you to type naïve.

In the past, \LaTeX has not supported input containing accented characters such as ñ, because Windows, Macintosh and Unix all have different ways of dealing with accented input, called input encodings.

However, the \inputenc package allows you to specify which input encoding your document is written with, for example to use the ISO Latin-1 encoding, you type:

\usepackage[latin1]{inputenc}

At the moment, \inputenc supports the ascii and latin1 input encodings, but more will be added with future releases.

The \inputenc package is currently a test release. The user interface for the full release will be upwardly compatible with the test version.

AMS-\LaTeX

AMS-\LaTeX is a set of miscellaneous extensions for \LaTeX distributed by the American Mathematical Society. They provide superior information structure and superior printed output for mathematical documents.

There are far too many features of AMS-\LaTeX to list here. AMS-\LaTeX is described in the accompanying documentation, and in The \LaTeX Companion.

Version 1.2beta of AMS-\LaTeX was released for testing by intrepid users in October 1994. The full release of AMS-\LaTeX 1.2 is expected in early January 1995.

It will be divided into two bundles:

- the \amsfonts packages, which give access to hundreds of new mathematical symbols, and new math fonts such as blackboard bold and fraktur.
- the \amsmath packages, which provide finer control over mathematical typesetting, such as multi-line subscripts, enhanced theorem and proof environments, and improved displayed equations.

For compatibility with older documents, an \amstex package will be provided.

\LaTeX on the internet

\LaTeX has its own home page on the World Wide Web, with the URL:

http://www.tex.ac.uk/CTAN/latex/

This page describes \LaTeX and the \LaTeX3 project, and contains pointers to other \LaTeX resources, such as the user guides, the \TeX Frequently Asked Questions, and the \LaTeX bugs database.

The electronic home of anything \TeX-related is the Comprehensive \TeX Archive Network (CTAN). This is a network of cooperating ftp sites, with over a gigabyte of \TeX material:

ftp://ftp.tex.ac.uk/tex-archive/
ftp://ftp.shsu.edu/tex-archive/
ftp://ftp.dante.de/tex-archive/

For more information, see the \LaTeX home page.

Further information

For more information on \TeX and \LaTeX, get in touch with your local \TeX Users Group, or the international \TeX Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.
Welcome to \LaTeX{} News 3

An issue of \LaTeX{} News will accompany every future release of \LaTeX{}. It will tell you about important events, such as major bug fixes, newly available packages, or any other \LaTeX{} news.

June 1995 release of \LaTeX{}

June 1995 sees the third release of \LaTeX{}2\epsilon. We are on schedule to deliver a release of \LaTeX{} every six months, in December and June.

In the last \LaTeX{} News, we said “we don’t expect so much activity in the next six months,” which has turned out not to be true!

Additional input encodings

In the last release of \LaTeX{} we distributed a test version of the inputenc package which allows the use of input characters other than just a–z and A–Z. The package has proved to be robust, so we are now distributing an expanded version. The new release comes with a number of input encodings:

- ascii the standard encoding,
- latin1 the ISO Western European alphabet,
- latin2 the ISO Eastern European alphabet,
- cp437 the IBM codepage 437,
- cp850 the IBM codepage 850, and
- applemac the Apple Macintosh encoding.

These can be used by specifying an option to the inputenc package, for example:

\begin{verbatim}
\usepackage[latin1]{inputenc}
\end{verbatim}

The new input encodings are currently being tested, but we don’t expect any major changes.

\LaTeX{} getting smaller

In the past releases of \LaTeX{}2\epsilon, the amount of memory \LaTeX{} requires has increased, but we are pleased to say that this trend has been reversed. We hope that future releases of \LaTeX{} will continue to get smaller.

For example, on this document, the December 1994 release used 52,622 words of memory, and the June 1995 release uses 51,216 words of memory, which is a 2.7% reduction.

We are currently experimenting with other ways of reducing the size of \LaTeX{}. For example, we are experimenting with an option to remove the picture and tabbing environments from the \LaTeX{} kernel, and to load them from a file the first time they are used. This should help \LaTeX{} to run on machines with limited memory. See autoload.txt for details.

Distribution and modification

One topic of discussion that has kept us busy is the distribution and modification conditions of \LaTeX{}. We are committed to keeping \LaTeX{} as free reliable software, and ensuring that (as far as possible) \LaTeX{} documents will produce the same results on all systems.

The modification conditions are currently under discussion, and we would like to hear from anyone interested. Please read modguide.tex for more information.

AMS-\LaTeX{} full release

The AMS-\LaTeX{} packages were still in beta test in the December 1994 release of \LaTeX{}, and the full release came out in January 1995.

AMS-\LaTeX{} is described in the User’s Guide (amsldoc.tex) and in The \LaTeX{} Companion.

PostScript fonts

There is a new test release of the PSNFSS packages for accessing PostScript fonts in \LaTeX{}2\epsilon. This includes an update to all of the fonts, to remove many of the underfull and overfull \hbox warnings, and improve the setting of non-English languages.

The new release of \LaTeX{} removes all of the “hidden” uses of Computer Modern mathematics. For example, the footnote markers used to use math mode, so always used Computer Modern digits rather than ones from the current text font. This has now been fixed.

Further information

For more information on \TeX{} and \LaTeX{}, get in touch with your local \TeX{} Users Group, or the international \TeX{} Users Group, P. O. Box 869, Santa Barbara, CA 93102-0869, USA, Fax: +1 805 963 8358, EMail: tug@tug.org.

The \LaTeX{} home page is http://www.tex.ac.uk/ctan/latex/ and contains links to other WWW resources for \LaTeX{}.
Welcome to \LaTeX News 4

An issue of \LaTeX News will accompany every future release of \LaTeX. It will tell you about important events, such as major bug fixes, newly available packages, or any other \LaTeX news. This issue accompanies the fourth release of \LaTeX2ε.

\LaTeX getting smaller

The last release in June started a trend of \LaTeX becoming smaller, we are pleased to announce that this has continued with this release. In particular the experimental ‘autoload’ version described in autoload.txt is much smaller as more parts of \LaTeX are autoloaded.

New ‘concurrent’ docstrip

The time taken to ‘unpack’ this release from the documented sources should be much reduced (roughly half the time, depending on installation conditions). This is due to an improved version of the docstrip program that has been contributed by Marcin Woliński. This can write up to 16 files at once. The previous version could only write one file at a time which meant that it was very slow when producing many small files from the same source file as the source needed to be re-read for each file written.

New T1 encoded fonts

This year Jörg Knappen has completed a new release of the ‘Cork’ (T1) encoded Computer Modern fonts: the dc fonts release 1.2.

This release of the dc fonts fixes many bugs (including the missing ‘¿’ (¿) and ‘¡’ (¡) ligatures) and improves the fonts in many other ways. It is strongly recommended that you upgrade as soon as possible if currently you are using the old dc fonts, release 1.1 or earlier. The new fonts are available from the CTAN archives, in tex-archive/fonts/dc.

The names of the font files are different. This does not affect \LaTeX documents but does affect the installation procedure as it assumes that you have the new fonts, and will write suitable ‘fd’ files for those fonts. If you have not yet upgraded your dc fonts then, after unpacking the distribution, you must latex olddc.ins to produce ‘fd’ files for the old dc fonts. This must be done before the format is made. Running the test document at ltxcheck.tex the end of the installation will inform you if the wrong set of ‘fd’ files has been installed.

Note that this change does not affect the standard ‘OT1’ Computer Modern fonts that \LaTeX uses by default.

More robust commands

The commands \cite and \sqrt are now robust.

Although most commands with optional arguments are fragile, as documented, such commands defined using the second optional argument of \newcommand and its derivatives are now robust.

New Interface to building ‘extension’ classes

The mechanism provided by \DeclareOption, \ProcessOptions and \LoadClass has proved to be a powerful and expressive means of defining one class in terms of another ‘base’ class. However there have been some requests to simplify the declaration of the common case where you want the ‘base’ class to be called with all the options that were specified to the extension class. This is now provided by the new command \LoadClassWithOptions. A similar command \RequirePackageWithOptions is provided for package use. More details of this feature are provided in clsguide.tex and ltclass.dtx.

More Input Encodings

The experimental inputenc package allows a more natural style of input of accented and other characters. Three new input encodings are now supported.

- ansinew the Windows ansi encoding, as used in Microsoft Windows 3.x.
- cp437de a variant of cp437, which uses β rather than β in the appropriate slot.
- next the encoding used on Next computers.

Further information

For more information on \TeX and \LaTeX, get in touch with your local \TeX Users Group, or the international \TeX Users Group, 1850 Union Street, #1637, San Francisco, CA 94123, USA, Fax: +1 415 982 8559, EMail: tug@tug.org. The \LaTeX home page is http://www.tex.ac.uk/ctan/latex/ and contains links to other WWW resources for \LaTeX.
Welcome to \LaTeX\ News 5

This issue of \LaTeX\ News accompanies the fifth release of the new standard \LaTeX, \LaTeX\ 2\epsilon.

Extra possibilities for section headings

Most \LaTeX\ sectioning commands are defined using \texttt{\@startsection}. For example, the \texttt{article} class defines:

\begin{verbatim}
\newcommand\section{\@startsection
 {section}{1}{0pt}{-3.5ex plus-1ex minus-.2ex}%
 {2.3ex plus.2ex}{\normalfont\Large\bfseries}}
\end{verbatim}

The last argument specifies the style in which the section heading is to be typeset.

The new feature added at this release is that at the end of this argument you may specify a command that takes an argument. This command will be applied to the section number and heading. For example, one could use the \texttt{\MakeUppercase} command to produce uppercase headings.

A package or class file could contain:

\begin{verbatim}
\renewcommand\section{\@startsection
 {section}{1}{0pt}{-3.5ex plus-1ex minus-.2ex}%
 {2.3ex plus.2ex}{\normalfont\Large\MakeUppercase}}
\end{verbatim}

to produce section headings using uppercase medium weight text, rather than the bold text used by \texttt{article}. Note that, like the font choice, the uppercasing applies only to the actual heading (including any automatically generated section number), not to the text as it may appear in the running head or table of contents.

The ‘openany’ option in the ‘book’ class

The \texttt{openany} option allows chapter and similar openings to occur on left hand pages. Previously this option only affected \texttt{\chapter} and \texttt{\backmatter}. It now also affects \texttt{\part}, \texttt{\frontmatter} and \texttt{\mainmatter}.

More font (output) encodings

The font encoding name \texttt{T3} has been allocated to the encoding used in the new 256-character IPA fonts (for the phonetic alphabet) produced by Rei Fukui. His package, \texttt{tipa}, gives access to these fonts and should soon be available. (The encoding named \texttt{OT3} is the 128-character encoding used in the IPA fonts produced by Washington State University.)

More input encodings supported

The \texttt{inputenc} package now supports the IBM codepage 852 used in Eastern Europe, with the option \texttt{[cp852]} contributed by Petr Sojka.

Also, the \texttt{inputenc} package now activates most ‘control codes’ with ASCII values below 32. Currently none of the encodings in the standard distribution makes use of these positions.

Fixes and improvements

The \LaTeX\ kernel has only had minor changes, apart from \texttt{\@startsection} mentioned above. However, some small fixes have been incorporated removing the following problems:

- In tabular and array, previous versions of \LaTeX\ ‘lost’ the inter-column space from an ‘1’-column, when that column was completely empty.
- Previously, the use of the \texttt{\nofiles} command could change the vertical spacing in a document. A side effect of fixing this is that when \texttt{\nofiles} is used, \texttt{\label} puts a blank line in the log file.
- \LaTeX\ often loads fonts ‘on demand’. Previously, this could happen inside the argument of an accent command and this would cause the accent to appear in the wrong place.

Changes to the ‘tools’ packages

- The \texttt{longtable} package now uses a modified algorithm, contributed by David Kastrup, to align the ‘chunks’ of a table. It is now unnecessary to edit the document to add \texttt{\setlongtables} before the final run of \LaTeX. In certain cases of overlapping \texttt{\multicolumn} entries, the new algorithm will produce better column widths than the old (at the price of extra passes through \LaTeX).
- The \texttt{dcolumn} package now has the extra possibility of specifying the number of digits both before and after the ‘decimal point’. This makes it easy to centre the column of numbers under a wide heading.

New copy of the \LaTeX\ bug database

http://www.tex.ac.uk/ctan/latex/bugs.html will soon have links to a copy of the searchable \LaTeX\ bugs
database at Mainz (Germany) as well as the original copy at Sussex (England).
Welcome to \LaTeX{} News 6

This issue of \LaTeX{} News accompanies the sixth release of the new standard \LaTeX{}, \LaTeX{}2\epsilon{}.

Mono-case file names

Previously \LaTeX{} has used some files with ‘mixed-case’ file names such as \texttt{T1cmr.fd} and \texttt{T1enc.def}.

These file names cause problems on some systems (in particular they are illegal on the ISO 9660 CDROM format) and so in this release all file names have been made lowercase (for example \texttt{t1cmr.fd} and \texttt{t1enc.def}).

This change should \textit{not} affect any document. Within \LaTeX{}, encodings still have the usual uppercase names in uses such as \texttt{\usepackage[T1]{fontenc}} and \texttt{\fontencoding{T1}}. \LaTeX{} will automatically convert to the lowercase form while constructing the file name. \LaTeX{} will input the ‘fd’ file under the old name if it fails to find the file with the new name, so existing collections of fd files should still work with this new release.

The change \textit{does} affect the configuration files that may be used to make the \LaTeX{} format with initex. For example, the file \texttt{fonttext.ltx} previously specified \texttt{\input{T1cmr.fd}}. It now has \texttt{\input{t1cmr.fd}}. If you use a local file \texttt{fonttext.cfg} you will need to make similar changes, as \texttt{\input{T1cmr.fd}} will not work as \texttt{T1cmr.fd} is no longer in the distribution.

The files affected by this change all have names of the form \texttt{*fd} or \texttt{*enc.def}.

Another input encoding

Thanks to work by Søren Sandmann, the inputenc package now supports the IBM codepage 865 used in Scandinavia.

Better user-defined math display environments

Suppose that you want to define an environment for displaying text that is numbered as an equation. A straightforward way to do this is as follows:

\begin{verbatim}
\newenvironment{texteqn}
{\begin{equation}
 \begin{minipage}{0.9\linewidth}
 {0.9\linewidth}}
{\end{minipage}}
\end{equation}
\end{environment}
\end{verbatim}

However, if you have tried this then you will probably have noticed that it does not work perfectly when used in the middle of a paragraph because an inter-word space appears at the beginning of the first line after the environment.

There is now an extra command (with a very long name) available that you can use to avoid this problem; it should be inserted as shown here:

\begin{verbatim}
\newenvironment{texteqn}
{\begin{equation}
 \begin{minipage}{0.9\linewidth}
 {0.9\linewidth}}
{\end{minipage}}
\end{equation}
\ignorespacesafterend}
\end{environment}
\end{verbatim}

Docstrip improvements

The docstrip program that is used to unpack the \LaTeX{} sources has undergone further development. The new version should be able to process all old ‘batchfiles’ but it allows a simpler syntax in new ‘batchfiles’ (no need to define \texttt{\def\batchfile{...}).

It also allows ‘target’ directories to be specified when writing files. This directory support is disabled by default unless activated in a local \texttt{docstrip.cfg} configuration file. See \texttt{docstrip.dtx} for details.

AMS \LaTeX{} update

Since the last \LaTeX{} release in June, the American Mathematical Society have re-issued the ‘AMSL\LaTeX{}’ classes and packages, fixing several reported problems.

Graphics package update

The \LaTeX{} color and graphics packages have been updated slightly, principally to support more dvi drivers, see the readme file in the \texttt{graphics} distribution.

EC Fonts released

The first release of the Extended Computer Modern fonts has just been made. (In January 1997.)

This release of \LaTeX{} does \textit{not} default to these ‘ec’ fonts as its T1 encoded fonts. By default it will use the ‘dc’ fonts if the T1 encoding is requested.

As noted in \texttt{install.txt} you may run \TeX{} on the install file \texttt{ec.ins} after unpacking the base distribution but \textit{before} making the \LaTeX{} format. This will produce suitable ‘fd’ files making \LaTeX{} (including, for the first time, the slides class) use the ‘ec’ fonts as the default T1 encoded font set.
T1 encoded Computer Modern fonts
As in the last release the base \LaTeX{} distribution contains three different sets of `fd' files for T1 encoded fonts.
In this release the default installation uses \texttt{ec.ins} and so installs files suitable for the current `EC fonts' distribution. If you have still not updated to the EC fonts and are using the earlier test versions, known as DC then you should unpack \texttt{newdc.ins} (for DC release 1.2 or later) or \texttt{olddc.ins} (for the original releases of the DC fonts). This should be done after unpacking \texttt{unpack.ins} but before making the format by running \texttt{ini\LaTeX{}} on \texttt{latex.ltx}. There are further details in \texttt{install.txt}.

T1 encoded Concrete fonts
The Metafont sources for T1 encoded ‘Concrete’ fonts have been removed from the \texttt{mfnss} distribution as they were based on the now obsolete DC fonts release 1.1. Similarly the \texttt{cmextra.ins} install file in the base distribution no longer generates fd files for the ‘Concrete’ fonts. To use these fonts in either T1 or OT1 encoding it is recommended that you obtain Walter Schmidt’s \texttt{ccfonts} package and fonts from CTAN \texttt{macros/latex/contrib/supported/ccfonts}.

Further input encodings
Two more \texttt{inputenc} packages have been added: for latin5, thanks to H. Turgut Uyar; and for latin3, thanks to Jörg Knappen.

Normalising spacing after punctuation
The command \texttt{\normalisecodes} was introduced at the last patch release. This is normally given the correct definition automatically and so need not be explicitly set. It is used to correct a problem, reported by Donald Arseneau, that punctuation in page headers has always (in all known \TeX{} formats) been potentially incorrect if the page break happens while a local setting of the space codes (for instance by the command \texttt{\frenchspacing}) is in effect. A common example of this happening in \LaTeX{} is in the \texttt{verbatim} environment.

Accessing Bold Math Symbols
The \texttt{tools} distribution contains a new package, \texttt{bm}, which defines a command \texttt{\bm} that allows individual bold symbols to be accessed within a math expression (in contrast to \texttt{\boldmath} which makes whole math expressions default to bold fonts). It is more general than the existing \texttt{amsbsy} package; however, to ease the translation of documents between these two packages, \texttt{bm} makes \texttt{\boldsymbol} an alias for \texttt{\bm}.
This package was previously made available from the ‘contrib’ area of the CTAN archives, and as part of Y&Y’s \LaTeX{} support for the MathTime fonts.

Policy on standard classes
Many of the problem reports we receive concerning the standard classes are not concerned with bugs but are suggesting, more or less politely, that the design decisions embodied in them are ‘not optimal’ and asking us to modify them.
There are several reasons why we have decided not to make such changes to these files.
• However misguided, the current behaviour is clearly what was intended when these classes were designed.
• It is not good practice to change such aspects of ‘standard classes’ because many people will be relying on them.

We have therefore decided not to even consider making such modifications, nor to spend time justifying that decision. This does not mean that we do not agree that there are many deficiencies in the design of these classes, but we have many tasks with higher priority than continually explaining why the standard classes for \LaTeX{} cannot be changed.
We would, of course, welcome the production of better classes, or of packages that can be used to enhance these classes.

New addresses for TUG
For information about joining the \TeX{} Users Group, and about lots of other \LaTeX{}-related matters, please contact them at their new address:
\TeX{} Users Group, P.O. Box 1239, Three Rivers, CA 93271-1239, USA
Fax: +1 209 561 4584
E-mail: tug@mail.tug.org
URL: http://www.tug.org/
New supported font encodings

Two new font encodings are supported as options to the fontenc package:

**OT4** This is a seven-bit encoding designed for Polish. The \LaTeX{} support was developed by Mariusz Olko.

**TS1** This is the ‘Text Companion Encoding’: it contains symbols designed to be used in text, as opposed to mathematical formulas, and some accents designed for uppercase letters. It is currently supported by the ‘tc’ fonts, which match the T1 encoded ‘ec’ text fonts. A subset of the glyphs in this encoding is supported by virtual fonts distributed with the PostScript font metrics on the ctan archives. (This is the ‘8c’ encoding in Karl Berry’s fontname scheme.) The textcomp package provides access to this encoding but here is a warning to current users of that package: some of the internal names for the characters have changed.

New input encodings

These additions to the inputenc package are decmulti (the DEC Multinational Character Set, contributed by M. Y. Chartoire) and cp1250 (an MS-Windows encoding for Central and Eastern Europe, contributed by Marcin Woliński). There is also a cp1252 encoding that is identical to ansinew.

Tools

The calc package (used in many examples in The \LaTeX{} Companion) has been contributed to this distribution by Kresten Krab Thorup and Frank Jensen. This is essentially the same as the version that has been available from the CTAN archives for some time, with one minor change: to use \LaTeX{}-style error messages. It enables the use of arithmetic expressions within arguments to standard \LaTeX{} commands where a length or a counter value is required. For example:

\begin{verbatim}
\setcounter{page}{\value{page} \* 2 + 1}
\parbox{3in - (2mm + \textwidth/9)}
\end{verbatim}

There have also been some improvements to several other packages in this collection. In particular, bm now works correctly with constructions such as \texttt{$bm{f'}$} involving ‘ or other characters which use \TeX{}’s special “\texttt{\mathcode='8000}” feature. Also, multicol sets the length \texttt{\columnwidth} to an appropriate value; this enables it to work with classes that support two-column setting, e.g., the AMS classes.

Graphics

The special oztex.def driver file has been removed, and Oz\TeX{} support has been merged with dvips, following advice from Andrew Trevorrow about Oz\TeX{} 3.x.

The keyval package has had some internal improvements: to use \LaTeX{} format error messages; and to avoid ‘# doubling’. This latter change means that the command key for the graphicx version of \texttt{\includegraphics} should now be used with one # rather than two. For example, \texttt{command = ‘gunzip #1}. Fortunately this key is almost never used in practice, so few if any documents should be affected by this change.

\LaTeX{}3 experimental programming conventions

As announced at the \TeX{} Users Group meeting (Summer 1997), a group of highly experimental packages will soon be released to allow experienced \TeX{} programmers to experiment with, and comment on, a proposed set of syntax conventions and basic data-types that might form the basis for programming large scale projects in \TeX{}. They will be located in this CTAN directory:

\begin{verbatim}
CTAN:macros/latex/packages/expl3
\end{verbatim}

The documentation of this material is as follows: individual package files provide outline, draft documentation; there is an article that gives an overview of the syntax and related concepts; there is a readme.txt file containing a brief description of the collection.

All aspects of these packages are liable, indeed likely, to change. They should not be used at this stage for anything that requires a stable system. However, we do encourage people to experiment with these packages, and to send comments on them to the \LaTeX-L mailing list. To subscribe to this list, mail to:

\begin{verbatim}
listserv@urz.uni-heidelberg.de
\end{verbatim}

the following one line message:

\begin{verbatim}
subscribe LATEX-L <first-name> <second-name>
\end{verbatim}
New math font encodings
A joint working group of the TeX Users Group and the \LATEX{} Project is developing a new 8-bit math font encoding for \TeX{}. It is designed to overcome several limitations and implementation problems of the old math font encodings and to simplify switching between different sets of math fonts, much as the \LATEX{} font selection interface has simplified switching between text fonts.

Since the work on this project relies entirely on volunteer work, we cannot give a specific release date yet. However, a prototype implementation already exists. This contains several sets of virtual fonts, some \LATEX{} packages and a kernel module; we hope to integrate it into the main \LATEX{} distribution for the next release.

Documents using only standard \LATEX{} commands for math symbols should not be affected by switching to the new math font encodings. However, documents, classes or packages making specific assumptions about the encoding of math symbol fonts are likely to break.

Further information about the Math Font Group may be found on the World Wide Web at http://www.tug.org/twg/mfg/.

A new math accent
A new math accent, \textbackslash{mathring}, has been added. This is a math mode version of the ring accent (˚) which is available in text mode with the command \textbackslash{r}.

Extended \textbackslash{DeclareMathDelimiter}
The command \textbackslash{DeclareMathDelimiter} has been extended. Normally this command takes six arguments. Previously, when being used to declare a character (such as \textbackslash{[}) as a delimiter, a variant form was used with only five arguments. The argument specifying the default ‘math class’ was omitted. Now the full six-argument form may be used in this case. The extra information is used to implicitly declare the character via \textbackslash{DeclareMathSymbol} for use when the symbol is not used with \textbackslash{left} or \textbackslash{right}.

The old five-argument form is detected and will work as before.

Tools distribution
The \texttt{multicol} package now supports the production of multiple columns without balancing the last page. To get this effect use the \texttt{multicol*} environment.

The \texttt{layout} package was partly recoded by Hideo Umeki to display page layout effects in a better way.

As suggested by Donald Arseneau, the calc package was extended to support the new commands \texttt{\widthof{text}}, \texttt{\heightof{text}}, and \texttt{\depthof{text}} within a \texttt{calc}-expression. At the same time we modified a few kernel commands so that \texttt{calc}-expressions can now be used in various useful places such as the dimension arguments to the \texttt{tabular} environment and the \texttt{\rule} command. For many other standard \LATEX{} commands this was already possible.

Support for Cyrillic encodings
We are very pleased that, after a lengthy period of development, a set of fonts, encodings and support files for using \LATEX{} with Cyrillic characters will soon be available.

Test versions of the ‘LH’ fonts for these Cyrillic encodings, based on the Computer Modern design, are available from CTAN archives in the directory fonts/cyrillic/lh-test. The \LATEX{} support files (by Werner Lemberg and Vladimir Volovich) are also available from CTAN archives in macros/latex/contrib/supported/t2

Default docstrip header
Many \LATEX{} users now distribute packages in documented source form using the docstrip system. Docstrip allows a header to be placed on generated package files, suitable for giving copyright information, or distribution conditions.

We have changed the default version of this header so that it allows stripped files to be distributed in ready-to-run installations such as the \TeX{}Live CD. If you use the default header for distributing your files you should check that the new copyright text is acceptable to you. The file docstrip.dtx explains how to produce your own header if you wish to do so.
Five years of \LaTeX2ε

Since this is the 10th edition of \LaTeX News, the (no longer) New Standard \LaTeX must have hit the streets almost this long ago. In fact it was only the beta-version that some people got just in time for Christmas 1993, and since then there has been a lot of tidying-up and smoothing of rough edges (not to mention a few bug fixes!).

Maybe it is time for something more radically different to emerge and be hungrily adopted by the world; but don’t panic, we shall be maintaining what you have now for a long time yet. Amongst the more polite things that have been written about our efforts, we found that this quote (somewhat censored to protect the guilty) well reflects some of our feelings about working on \LaTeX over the years:

\begin{quote}
the mere existence of \LaTeX2ε is a great miracle.
\end{quote}

Restructuring the \LaTeX distribution

Since the (once) ‘new’ standard \LaTeX has reached such a venerable age, we are reviewing the way in which the system is presented to the world.

An early intention is to define, given the wide variety of good packages now available, what now constitutes a useful installation of \LaTeX. We also hope that such a definition will help document portability if it leads to a future in which a \LaTeX class designer can reasonably assume that a known list of facilities will be there for all users (so that each class need not supply them).

As a first small step towards this definition, we shall replace the \texttt{latex/packages} subdirectory on CTAN. This directory was a curious mixture of the important, such as the \LaTeX tools, that any self-respecting \LaTeX installation ought to have, and the esoteric or experimental.

The esoterica from \texttt{packages} will be moved to new locations, as follows:

\begin{verbatim}
expl3 to latex/exptl/project
mfnfss to latex/contrib/supported/mfnfss
\end{verbatim}

The subdirectory that replaces \texttt{packages} will be called \texttt{latex/required}; all the other sub-directories of \texttt{packages} will be moved there.

\LaTeX Project on the Internet

A new \texttt{latex-project.org} domain has been registered. The web site is not yet fully functional but the old \LaTeX pages from CTAN are available at \url{http://www.latex-project.org/} and the \LaTeX bug reporting address has been changed to \texttt{latex-bugs@latex-project.org}.

Restructuring the \LaTeX package licenses

Several people have requested an easy mechanism for the distribution of \LaTeX packages and other software “under the same conditions as \LaTeX”. The old \texttt{legal.txt} file was unsuitable as a general licence as it referred to specific \LaTeX authors, and to specific files.

Therefore, in this release \texttt{legal.txt} contains just the copyright notice and a reference to the new \LaTeX Project Public License (LPPL) for the distribution and modification conditions. The \texttt{tools}, \texttt{graphics}, and \texttt{mfnfss} packages also now refer to this license in their distribution notices.

Support for Cyrillic encodings

Basic Cyrillic support, as announced in \LaTeX News 9, is now finally an official part of \LaTeX. It includes support for the following standard Cyrillic font encodings (this list may grow): T2A T2B T2C X2.

It also includes various Cyrillic input encodings (20 in total, including commonly used variants and Mongolian Cyrillic encodings). This provides platform independent and sophisticated basic support for high-quality typesetting in various Cyrillic-based languages.

For further information see the file \texttt{cyrguide.tex}.

Tools distribution

The \texttt{varioref} package has been extended to support textual page references to a range of objects: e.g., if \texttt{eq-first} and \texttt{eq-last} are the label names for the first and last equation in a sequence, then you can now write

\begin{verbatim}
see\-\vrefrange{eq-first}{eq-last}
\end{verbatim}

This results in different text depending on whether both labels fall on the same page.

Some additional user commands, as well as building-blocks for writing private extensions, are described in the accompanying documentation.
Back in sync

The last release of \LaTeX{} was delayed even longer than you have come to expect. We hope that it proved worth waiting for. It required a major integration of the code from several people and, independently, the introduction of the LPPL (see \LaTeX{} News 10) plus several related changes to our internal systems. It therefore seemed sensible to wait until everything was complete rather than do things in too much hurry.

This seems to have been a successful strategy as the recent patch release was related to an isolated change that was done many months previously. If this release does not appear a lot closer to its nominal date then ... well, you will not be reading this sentence!

Yearly release cycles

With the year 2000 rapidly approaching, we intend to switch to a release frequency of just one per year (with patches if necessary) for the core of \LaTeX{}\textsc{2}\textepsilon. These days the system is sufficiently stable that the original update policy is costing everybody more time than is now warranted.

LPPL update

Thanks to extensive and valuable input from Matt Swift (swift@alum.mit.edu) we now have a clearer and more detailed form of the \LaTeX{} Project Public Licence. This release contains both the original version (in \texttt{lppl-1-0.txt}) and the updated version, LPPL 1.1.

The future of Sli\TeX{}

We still get a very small trickle of reports about this part of the system (if you are no longer able to recall \LaTeX{} 2.09 then you will know it as the \texttt{slides} class). We have not classified them (in our minds at least) as bugs since we have always known that there are many problems with this class. It is clear to us that the only sensible action would be to redesign the system completely; in particular, to remove much of its complexity whose purpose is to support 10-year-old overlay technology. However, this would take a lot too much time and would be completely out of proportion to its current usage.

We are therefore planning to make the \texttt{slides} class unsupported in the sense that any problem related to the use of invisible fonts is considered to be a feature (The \LaTeX{}\textsc{2}\textepsilon~manual by Leslie Lamport doesn't even describe this part of the class any more). Of course, if it still has its enthusiasts then we are happy to cede it to their loving care (somewhat like a preserved steam locomotive, in some parts of the world).

Fontenc package peculiarities

The \texttt{usepackage} interface normally ensures that a package is loaded only once. The \texttt{fontenc} package has become an exception to this rule: it can be loaded several times using different options, e.g., allowing the user to add a font encoding in the preamble. This comes at a price for package writers: the low-level commands (see \texttt{ltclass.dtx}) used to check if a package was loaded, and with which options, do not work for the \texttt{fontenc} package.

New math font encodings

As we announced in \LaTeX{} News 9, a joint working group of the \TeX{} Users Group and the \LaTeX{} Project has developed a new 8-bit math font encoding for \TeX{}. The reason why this work is not yet released is because of other exciting developments in the world of math fonts and math characters. It is obviously wise to ensure that the encoding work is fully integrated with the available fonts.

Those interested are reminded that further information about the Math Font Group may be found on the World Wide Web at: \url{http://www.tug.org/twg/mfg/}.

Tools distribution

The \texttt{multicol} package has now got a small but useful extension which allows you to force a column break where this is really necessary. This is done with the command \texttt{\columnbreak}, which can be used like \texttt{\pagebreak} (e.g., within paragraphs) except that it cannot have an optional argument and thus it always forces a new column.

Coming soon

Major work on a new class file structure to support flexible designs is well under way; some of this work will be presented at the TUG’99 conference in Vancouver, Canada. With a bit of luck much of this work could be ready for integration into the next release—so watch this space!
**LPPL update**

Since the release of the \LaTeX\ Project Public Licence version 1.1, we have received a small number of queries which resulted in some minor changes to improve the wording or explain the intentions better. As a consequence this release now contains LPPL 1.2 in the file `lppl.txt` and the previous versions as `lppl-1-0.txt` and `lppl-1-1.txt`.

**fixltx2e package**

This package provides fixes to \LaTeX\ε which are desirable but cannot be integrated into the \LaTeX\ε kernel directly as they would produce a version incompatible to earlier releases (either in formatting or functionality).

By having these fixes in the form of a package, users can benefit from them without the danger that their documents will fail, or produce unexpected results, at other sites; this works because a document will contain a clear indication (the `\usepackage` line, preferably with a required date) that at least some of these fixes are required to format it.

**Outcome of TUG '99 (Vancouver)**

The slides from the TUG'99 presentation we gave on a new interface for \LaTeX\ class designers are available from the \LaTeX\ Project website; look for the file `tug99.pdf` at:

\begin{verbatim}
http://www.latex-project.org/talks/
\end{verbatim}

Please note that this document was intended only to be informal “speaker’s notes” for our own use. We decided to make them available (the speaker’s notes as well as the slides that were presented) because several people requested copies after the talk. However, they are not in a polished copy-edited form and are not intended for publication.

Prototype implementations of parts of this interface are now available from:

\begin{verbatim}
http://www.latex-project.org/code/experimental/
\end{verbatim}

We are continuing to add new material at this location so as to stimulate further discussion of the underlying concepts. As of December 1, 1999 the following parts can be downloaded:

- **xpars**e Prototype implementation of the interface for declaring document command syntax. See the `.dtx` files for documentation.

- **template** Prototype implementation of the template interface (needs parts of `xpars`).

  The file `template.dtx` in that directory has a large section of documentation at the front describing the commands in the interface and giving a ‘worked example’ building up some templates for caption formatting.

- **xcontents** Interface description for table of contents data (no code yet). Coding examples have been thoroughly discussed on the \latex-l list.

- **xfootnote** Working examples for generating footnotes, etc. Needs `xpars` and `template`.

All examples are organised in subdirectories and additionally available as `gzip tar` files.

Please remember that this material is intended only for experimentation and comments; thus any aspect of it, e.g., the user interface or the functionality, may change and, in fact, is very likely to change. For this reason it is explicitly forbidden to place this material on CD-ROM distributions or public servers.

These concepts, as well as their implementation, are under discussion on the list \latex-l. You can join this list, which is intended solely for discussing ideas and concepts for future versions of \LaTeX, by sending mail to `listserv@URZ.UNI-HEIDELBERG.DE` containing the line

```
SUBSCRIBE \latex-l Your Name
```

This list is archived and, after subscription, you can retrieve older posts to it by sending mail to the above address, containing a command such as:

```
GET \latex-l LOGyy\mm
```

where `yy=Year` and `\mm=Month`, e.g.

```
GET \latex-l LOG9910
```

for all messages sent in October 1999.
Yearly release cycle

We announced in \LaTeX{} News 11 that we intended to switch to a 12-monthly release schedule. With the present (June 2000) release, this switch is being made: thus the next release of \LaTeX{} will be dated June 2001. We shall of course continue, as in the past, to release patches as needed to fix significant bugs.

PSNFSS: Quote of the Month

You should say in the \LaTeX{} News that Walter Schmidt has taken over PSNFSS from me. It gives me a certain pleasure to be able to draw a line under that part of my life…  

Sebastian Rahtz

The PSNFSS material, which supports the use of common PostScript fonts with \LaTeX{}, has been thoroughly updated. Most noticeably, the mathppl package, which used to be distributed separately, is now part of the basic PSNFSS bundle; this package provides mathematical typesetting with the Palatino typeface family. In addition, numerous bugs and flaws have been fixed and the distribution has been ‘cleaned up’. The file changes.txt contains a detailed list of these changes.

The documentation (in psnfss2e.pdf) has been completely rewritten to provide a comprehensive introduction to the use of PostScript fonts.

Notice that the new PSNFSS needs updated files for font metrics, virtual fonts and font definitions. If you received the new version (8.1) as part of a complete \TeX{} system then these new font files should also have been installed. However, if you intend to install or update PSNFSS yourself, please read the instructions in the file 00readme.txt of the new PSNFSS distribution.

Support for commercial PostScript fonts, such as Lucida Bright, has been removed from the basic distribution; it is now available from CTAN: http://mirror.ctan.org/macros/latex/contrib/supported/psnfssx.

New AMS-\LaTeX{}

Version 2.0 of AMS-\LaTeX{} was released on December 1, 1999. It can be obtained via ftp://ftp.ams.org/pub/tex/ or http://www.ams.org/tex/amslatex.html, as well from CTAN: http://mirror.ctan.org/macros/latex/required/amslatex.

This release consists chiefly of bug fixes and consolidation of the existing features. The division of AMS-\LaTeX{} into two main parts (the math packages; the AMS document classes) has been made more pronounced. The files diffs-m.txt, diffs-c.txt, amsmath.faq, and amsclass.faq describe the changes and address some common questions.

The primary documentation files remain amsldoc.tex for the amsmath package, and instr-l.tex for the AMS document classes. The documentation for the amsthm package, however, has been moved from amsldoc.tex to a separate document amsthdoc.tex.

New input encoding latin4

The package inputenc has, thanks to Hana Skoumalová, been extended to cover the latin4 input encoding; this covers Baltic and Scandinavian languages as well as Greenland Inuit and Lappish.

New experimental code

In \LaTeX{} News 12 we announced some ongoing work towards a ‘Designer Interface for \LaTeX{}’ and we presented some early results thereof. Since then, at Gutenberg2000 in Toulouse and TUG2000 in Oxford, we described a new output routine and an improved method of handling vertical mode material between paragraphs. In combination these support higher quality automated page-breaking and page make-up for complex pages—the best yet achieved with \TeX{}!

A paper describing the new output routine is at http://www.latex-project.org/papers/xo-pfloat.pdf All code examples and documentation are available at http://www.latex-project.org/code/experimental/. This directory has been extended to contain

galley Prototype implementation of the interface for manipulating vertical material in galleys.

xinitials Prototype implementation of the interface for paragraph initials (needs the galley package.

xtheorem Contributed example using the template package to provide a designer interface for theorem environments.

xoutput A prototype implementation of the new output routine as described in the xo-pfloat.pdf paper. Expected availability: at or shortly after the TUG2000 conference.

\footnote{The stress here is on automated!}
Future releases

We are currently exploring how to best support the very large community of individuals, organisations and enterprises that depend on the robustness and availability of the current standard \LaTeX{} distribution. The results of this may lead to some changes in the regular release schedule and the handling of bug reports during the next year.

New release of Babel (required)

Earlier this year a new release of Babel (3.7) became available. You can read about its new features in http://www.ctan.org/tex-archive/macros/latex/required/babel/announce.txt

One of the bugs that got fixed in this release deals with how labels are handled by \LaTeX{}. Because this part of the kernel is modified by babel, the relevant changes need to be coordinated. Therefore to use Babel with this release of \LaTeX{} you will need to update your version of babel to at least 3.7.

New input encoding latin9

The package inputenc has, thanks to Karsten Tinnefeld, been extended to cover the latin9 input encoding. The ISO-Latin 9 encoding is a useful modern replacement for ISO-Latin 1 that contains a few characters needed for French and Finnish. Of wider interest, it also contains the euro currency sign; this could be the killer argument for many 8-bit texts to use Latin-9 in the future.

According to a Linux manpage, ISO Latin-9 supports Albanian, Basque, Breton, Catalan, Danish, Dutch, English, Estonian, Faroese, Finnish, French, Frisian, Galician, German, Greenlandic, Icelandic, Irish Gaelic, Italian, Latin, Luxembourgish, Norwegian, Portuguese, Rhaeto-Romanic, Scottish Gaelic, Spanish and Swedish. The characters added in latin9 are (in \LaTeX{} notation):

\texteuro \textbsy S \textbsy s \textbsy Z \textbsy z \OE \OE \textasciidcy Y

They displace the following characters from latin1:

\textcurrency \textbrokenbar \textonequarter \textonehalf \textthreequarters

New tools

The new package trace provides many commands to control \LaTeX{}'s tracing and debugging output, including the excellent new information available with \TeX{} such as the extremely useful tracing of local assignments. You will find it in the tools distribution.

\texteuro\textbsy S\textbsy s\textbsy Z\textbsy z\OE\OE\textasciidcy Y

It offers the command \texttt{\textbackslash traceon}, which is similar to \texttt{\textbackslash tracingall} but suppresses uninteresting stuff such as font loading by NFSS (which can go on for pages if you are unlucky). It also offers \texttt{\textbackslash traceoff} to ... guess what! Full details are in the documented source file, trace.dtx.

In the base ifthen package we have added the uppercase synonyms \texttt{\textbackslash NOT \textbackslash AND} and \texttt{\textbackslash OR}.

New experimental code

In \LaTeX{} News 12 we announced some ongoing work towards a ‘Designer Interface for \LaTeX{}’ and we presented some early results thereof. Since then, at Gutenberg2000 in Toulouse and TUG2000 in Oxford, we described a new output routine and an improved method of handling vertical mode material between paragraphs. In combination these support higher quality automated\textsuperscript{1} page-breaking and page make-up for complex pages—the best yet achieved with \TeX{}!

More recently we have added material to handle the complex front matter requirements of journal articles; this was presented at Gutenberg2001 in Metz.

A paper describing the new output routine is at http://www.latex-project.org/papers/xo-pfloat.pdf

All code examples and documentation are available at http://www.latex-project.org/code/experimental

This directory has been extended to contain the following.

galley Prototype implementation of the interface for manipulating vertical material in galleys.

xinitials Prototype implementation of the interface for paragraph initials (needs the galley package).

xtheorem Contributed example using the template package to provide a designer interface for theorem environments.

xor A prototype implementation of the new output routine as described in the xo-pfloat.pdf paper.

xfrontm A prototype version of the new font matter interface.

\textsuperscript{1}The stress here is on automated!
Anniversary release

Yes, it's now 10 years since the first release in this series and, for Knuthists, this release also contains Issue 16!

Meanwhile this Issue 15 describes the major new features in the current release whilst Issue 16 looks a little way into the future of \LaTeX.

LPPL – new version

Most importantly, there is now a new version, 1.3, of the \LaTeX\ Project Public Licence. Many of you will be thrilled to know that, following the exchange of over 1600 e-mail messages dissecting various aspects of its philosophy such as ‘how many angels can appear in the name of a file before it becomes non-free’, this version is now officially a DFSG (Debian Free Software Guidelines) approved license. The discussions start at http://lists.debian.org/debian-legal/2002/debian-legal-200207/threads.html with high traffic throughout August to October 2002 and further heated discussions starting in April 2003 and concluding around June at http://lists.debian.org/debian-legal/2003/debian-legal-200306/msg00206.html.

The important features of the new version are useful clarifications in the wording, and revised procedures for making a change to the Current Maintainer of a package. Special thanks to all those people from Debian Legal who worked constructively with us on this onerous task, especially but not exclusively Jeff Licquia and Branden Robinson.

Small updates to varioref

The English has been corrected in \texttt{\ref\texttt{textbefore}} (an incompatible change). There are other extensions such as \texttt{\labelformat}, \texttt{\Ref}, \texttt{\Vref} and \texttt{\vpagerefnum}. Some Dutch text has also been changed and two new options added: \texttt{slovak} and \texttt{slovene}.

New and more robust commands

Many of the math mode commands for compound symbols have been made robust and a new robust command has been added: \texttt{\nobreakdashes}. This last is a low-level command, borrowed from the \texttt{\amsmath} package, for use only before hyphens or dashes. It prevents the line break that is normally allowed after the following sequence of dashes.

Fixing font sizes

The new \texttt{fix-cm} package, by Walter Schmidt, changes the CM font definition (.fd) files so that similar design sizes are used in both the \texttt{OT1} and \texttt{T1} encodings.

Font encodings

A number of options have been added to the \texttt{textcomp} package, enabling only available glyphs to be used. Also, the ‘NFSS font families’ are now divided into five different groups according to the subset of glyphs each provides from the full collection of symbols in the TS1 encoding. Given sufficient information about a font family \texttt{textcomp} will use this in order to limit the typesetting to those glyphs that are available.

Use of this mechanism has also enhanced \texttt{\oldstylenums} to use the current font if possible.

Displaying font tables

With the \texttt{nffont} package you can now specify the font to display by giving its ‘NFSS classification’, rather than needing to know its external font file’s name. It is also now possible to generate large collections of font tables in batch mode by providing a suitable input file.

New input encodings

The \texttt{inputenc} package has been extended as follows: \texttt{macce} input encoding (Apple Central European), thanks to Radek Tryc and Marcin Wolinski; \texttt{cp1257} for Baltic languages; \texttt{latin10}, thanks to Ionel Ciobîcă. The euro symbol has by now been added to several encodings: \texttt{ansinew}, \texttt{cp1250} and \texttt{cp1252} (which also has another addition), whilst \texttt{cp858} adds it to \texttt{cp850}.

Unicode input

Partial, experimental support for text files that use the Unicode encoding form UTF-8 is now provided by the option \texttt{utf8} for the \texttt{inputenc} package.

The only Unicode text file characters supported by the current version are those based on the most common inputs for glyphs from the small collection of standard \LaTeX\ Latin encodings.

And finally ... \texttt{pict2e}

The old, non-functional version of this package has been removed as there is now a fully working version from Hubert Gäßlein and Rolf Niepraschk. It is described in \textit{The \LaTeX\ Manual}. 
**Anniversary news**

This anniversary Issue 16 takes a brief look into the future work of the \LaTeX{} Project Team, both short and longer range. Please let us know if you want to get involved with us in any of this work (see below).

An overview of the 10th Anniversary Release, dated 2003/12/01, is can be found in Issue 15.

**TLC2: The \LaTeX{} Companion – 2nd edition!**

Since you are reading this newsletter, there is a good chance that you, or a friend, has already bought this encyclopedic volume: the incomparable Second Edition of this work that is every \LaTeX{}e’s ultimate lucky charm.

If by some chance you have not yet purchased your own copy then get into training, get shopping, and get flexing your muscles (both physical—it’s 1100+ pages, and intellectual) by using it to discover masses of invaluable ‘insider information’ about:

- the latest release of Standard \LaTeX{};
- over 200 extension packages;
- plus related software and systems.

For more information on this all new (??...OK, not all, but over 90%!!), all accurate (we hope!) 10th Anniversary Edition, check out [http://www.awprofessional.com/titles/0201362996](http://www.awprofessional.com/titles/0201362996).

**Future maintenance**

We are currently exploring how best to support the very large and rapidly growing community of individuals, organisations and enterprises that depend on the robustness and availability of the current standard \LaTeX{} distribution. Although we remain firmly resolved not to make changes in the base distribution (the kernel) of Standard \LaTeX{}, there is still much that needs doing to maintain its reliability and utility and to keep up the necessary level of communication with users and supporters. Also, as with all advanced software systems, bugs are still turning up occasionally so some fixes are still essential.

One major impediment to providing adequate service levels in this area is, of course, the difficulties inherent in obtaining the time and commitment of skilled minds—hence the appeal above to anyone interested in getting involved.

**LPPL certification**

There are still some outstanding diplomatic tasks around the \LaTeX{} Project Public Licence: these include e.g., getting it ‘OSF certified’ and ensuring that it gains more support and wider use, even in the FSF world where it has long been tolerated.

**Use of \$\varepsilon\$-\TeX{}/pdf\TeX{}**

We expect that within the next two years, releases of \LaTeX{} will change modestly in order to run best under an extended \TeX{} engine that contains the \$\varepsilon\$-\TeX{} primitives, e.g., \$\varepsilon\$-\TeX{} or pdf\TeX{}. The details of this possible upgrade need further work so we are not making a definite announcement yet.

Although the current release does not require \$\varepsilon\$-\TeX{} features, we certainly recommend using an extended \TeX{}, especially if you need to debug macros.

**End of ’autoload’ support**

As computer systems generally grow in capacity, requirements change and so we believe that the autoload variant of \LaTeX{} is no longer required. Thus, although the code remains it is no longer supported. We hope this does not cause any problems.

**New models, new code**

In the period 1999–2001 we published many results of our work over the previous decade on the development of new concepts and models for automated typesetting based on \TeX{} as the underlying platform. These can be found at [http://www.latex-project.org/papers/](http://www.latex-project.org/papers/) and [http://www.latex-project.org/code/experimental/](http://www.latex-project.org/code/experimental/).

Since then a very large proportion of The Team’s efforts have been diverted to provide the core author team for TLC2, which provides over 1000 pages of carefully researched and tested documentation of many aspects of the vast world of \LaTeX{} related software that was developed over that time period and that continues to grow and improve prodigiously.

Completion of that task … until TLC3!! … presents the possibility of getting back to this more exciting development work, or even to more radical work on non-\TeX{}-based models and implementations.

Of course, any such ideas are predicated on our ability to organise (with you, we hope) an efficient but responsive maintenance and support system for Standard \LaTeX{}.
Project licence news

The \LaTeX{} Project Public License has been updated slightly so that it is now version 1.3c. In the warranty section the phrase “unless required by applicable law” has been reinstated, having got lost at some point. Also, it now contains three clarifications: of the difference between “maintained” and “author-maintained”; of the term “Base Interpreter”; and when clause 6b and 6d shall not apply.

Following requests, we now also provide the text of the licence as a \LaTeX{} document (in the file \texttt{lpl.tst}). This file can be processed either as a stand-alone document or it can be included (without any modification) into another \LaTeX{} document, e.g., as an appendix, using \texttt{\input} or \texttt{\include}.

New guide on font encodings

Way back in 1995 work was started on a guide to document the officially allocated \LaTeX{} font encoding names. However, for one reason or another this guide (named \texttt{\LaTeX{} font encodings}) was, until now, not added to the distribution. It describes the major 7-bit and 8-bit font encodings used in the \LaTeX{} world and explains the restrictions required of conforming text font encodings. It also lists all the ‘encoding specific commands’ (the LICR or \LaTeX{} Internal Character Representation) for characters supported by the encodings OT1 and OT2.

When the file \texttt{encguide.tex} is processed by \LaTeX{}, it will attempt to typeset an encoding table for each encoding it describes. For this to be possible, \LaTeX{} must be able to find .tfn files for a representative example font for each encoding. If \LaTeX{} cannot find such a file then a warning is issued and the corresponding table is omitted.

Robust commands in math

The font changing commands in text-mode have been robust commands for years, but the same has not been true for the math versions such as \texttt{\textbf{\LaTeX{}}} While the math-mode commands worked correctly in section heads, they could cause problems in other places such as index entries. With this release, these math-mode commands are now robust in the same way as their text-mode counterparts.

Updates of required packages

Several of the packages in the \texttt{tools} bundle have been updated for this release.

The \texttt{xspace} package has some new features. One is an interface for adding and removing the exceptions it knows about and another is that it works with active characters. These remove problems of incompatibility with the \texttt{blabel} system.

In \texttt{\LaTeX{} News 16} we announced that some packages might begin to take advantage of \varepsilon-\TeX{} extensions on systems where these are available; and the latest version of \texttt{xspace} does just that. Note also that \fixtxt2e will make use of the facilities in \varepsilon-\TeX{} whenever these are present (see below).

The \texttt{calc} package has also been given an update with a few extra commands. The commands \texttt{\maxof} and \texttt{\minof}, each with two brace-delimited arguments, provide the usual numeric max and min operations. The commands \texttt{\settowidth} and \texttt{\settopheight} work like \texttt{\settowidth} and \texttt{\settopheight}. There are also some internal improvements to make \texttt{calc} work with some more primitive \TeX{} constructs, such as \texttt{\ifcase}.

The \texttt{varioref} package has acquired a few more default strings but there are still a number of languages for which good strings are still missing.

The \texttt{showkeys} package has also been updated slightly to work with more recent developments in \texttt{varioref}. Also, it now provides an easy way to define the look of the printed labels with the command \texttt{\showkeyslabelformat}.

Work on \LaTeX{} fixes

The package known as \fixtxt2e has three new additions. A new command \texttt{\textsubscript{\texttt{text}}} has been added as a complement to the command \texttt{\textsuperscript{\texttt{text}}} in the kernel. Secondly, a new form of \texttt{\DeclareMathSizes} that allows all of its arguments to have a dimension suffix. This means you can now use expressions such as \texttt{\DeclareMathSizes{9.5dd}{9.5dd}{7.4dd}{6.6dd}}.

The third new addition is the robust command \texttt{\TextOrMath} which takes two arguments and executes one of them when typesetting in text or math mode respectively. This command also takes advantage of \varepsilon-\TeX{} extensions if available; more specifically, when the \varepsilon-\TeX{} extensions are available, it does not destroy kerning between previous letters and the text to be
typeset. The command is also used internally in \fnsymbol to resolve a problem with \fnsymbol.

Also, further work has been done on reimplementing the command \addpenalty, which is used internally in several places: we hope it is an improvement!

The graphics bundle

The graphics bundle now supports the dvipdfmx post-processor and Jonathan Kew’s XETeX program. By support we mean that the graphics packages recognize the new options xetex and dvipdfmx but we do not distribute the respective driver files.

This leads elegantly to a description of the new policy concerning such driver files in the graphics bundle. Most driver files for our graphics packages are maintained by the developers of the associated post-processor or \TeX programs. The teams developing these packages are working very hard: their rapid development offers a stark contrast to the current schedule of \TeX releases. It is therefore no longer practical for the \TeX Team to be responsible for distributing the latest versions of these driver files.

Therefore the installation files for graphics have been split: there is now graphics.ins to install the package files and graphics-drivers.ins for the driver files (located in drivers.dtx). There is no need to install all those provided in the file drivers.dtx.

Please also note that, as requested by the maintainers of PSTricks, we have removed the package pstcol as current versions of PSTricks make it obsolete.

Future development

The title of this section is a little misleading as it actually describes current development. In 1998 the expl3 bundle of packages was put on CTAN to demonstrate a possible \TeX3 programming environment. These packages have been lying dormant for some time while the \TeX Project Team were preoccupied by other things such as developing the experimental packages xor, template, etc., (and also writing that indispensable and encyclopaedic volume, The \TeX Companion – 2nd edition).

In October 2004 work on this code base was resumed with the goal of some day turning it into a kernel for \TeX3. This work can now also make full use of the widely accepted \v-\TeX extensions. Currently two areas are central to this work.

- Extending the kernel code of \TeX3.
- Converting the experimental packages such as xor, template to use the new syntax internally.

Beware! Development of expl3 is happening so fast that the descriptions above might be out of date when you read this! If you wish to see what’s going on then go to http://www.latex-project.org/code.html where you can download fully working code (we hope!).
This news never existed.
New LaTeX release

This issue of LaTeX News marks the first release of a new version of LaTeX since the publication of The LaTeX Companion in 2005–2006.

Just in time for TeX Live 2009, this version is a maintenance release and introduces no new features. A number of small changes have been made to correct minor bugs in the kernel, slightly extend the Unicode support, and improve various aspects of some of the tools packages.

New code repository

Since the last LaTeX release, the entire code base has been moved to a public svn repository and the entire build architecture re-written. In fact, it has only been possible for us to consider a new LaTeX release since earlier this year when the test suite was finally set up with the new system. In the process, a bug in the LaTeX picture fonts distributed with TeX Live was discovered, proving that the tests are working and are still very valuable.

Now that we can easily generate new packaged versions of the LaTeX distribution, we expect to be able to roll out bug fixes in a much more timely manner than over the last few years. New versions should be distributed yearly with TeX Live. Having said this, the maintenance of the LaTeX kernel is slowing down as the bugs become fewer and more subtle. Remember that we cannot change any of the underlying architecture of the kernel or any design decisions of the standard classes because we must preserve backwards compatibility with legacy documents at all costs.

Even new features cannot be added, because any new documents using them will not compile in systems (such as journal production engines) that are generally not updated once they’ve been proven to work as necessary.

None of this is to say that we consider LaTeX to be any less relevant for document production than in years past: a stable system is a useful one. Moreover, the package system continues to provide a flourishing and stable means for the development of a wide range of extensions.

Babel

One area of the LaTeX code base that is still receiving feedback to be incorporated into the main distribution is the Babel system for multilingual typesetting. While the Babel sources have already been added to the svn repository the integration of the test system for Babel is still outstanding.

The future

While work on LaTeX tends to maintenance over active development, the LaTeX3 project is seeing new life. Our goals here are to provide a transition from the LaTeX document processing model to one with a more flexible foundation. Work is continuing in the expl3 programming language and the packages for document design. Future announcements about LaTeX3 will be available via the LaTeX Project website and in TUGboat.

http://www.latex-project.org/svnroot/latex2e-public/
Scheduled \LaTeX{} bug-fix release

This issue of \LaTeX{} News marks the first bug-fix release of \LaTeX{}\texttt{2e} since shifting to a new build system in 2009. Provided sufficient changes are made each year, we expect to repeat such releases once per year to stay in sync with \TeX{} Live. Due to the excitement of \TeX{}’s 25-th birthday last year, we missed our window of opportunity to do so for 2010. This situation has been rectified this year!

Continued development

The \LaTeX{}\texttt{2e} program is no longer being actively developed, as any non-negligible changes now could have dramatic backwards compatibility issues with old documents. Similarly, new features cannot be added to the kernel since any new documents written now would then be incompatible with legacy versions of \LaTeX{}.

The situation on the package level is quite different though. While most of us have stopped developing packages for \LaTeX{}\texttt{2e} there are many contributing developers that continue to enrich \LaTeX{}\texttt{2e} by providing or extending add-on packages with new or better functionality.

However, the \LaTeX{} team certainly recognises that there are improvements to be made to the kernel code; over the last few years we have been working on building, expanding, and solidifying the \texttt{expl3} programming layer for future \LaTeX{} development. We are using \texttt{expl3} to build new interfaces for package development and tools for document design. Progress here is continuing.

Release notes

In addition to a few small documentation fixes, the following changes have been made to the \LaTeX{}\texttt{2e} code; in accordance with the philosophy of minimising forwards and backwards compatibility problems, most of these will not be noticeable to the regular \LaTeX{} user.

Font subsets covered by Latin Modern and \TeX{} Gyre

The Latin Modern and \TeX{} Gyre fonts are a modern suite of families based on the well-known Computer Modern and ‘PostScript 16’ families with many additional characters for high-quality multilingual typesetting.\footnote{See their respective TUGboat articles for more information: \url{http://www.tug.org/TUGboat/tpub/24-1/jackowski.pdf}}

Information about their symbol coverage in the TS1 encoding is now included in \texttt{textcomp}'s default font definitions.

Redefinition of \texttt{\end{document}} Inside the definition of \texttt{\end{document}} the \texttt{.aux} file is read back in to resolve cross-references and build the table of contents etc. From 2.09 days this was done using \texttt{\input} without any surrounding braces which could lead to some issues in boundary cases, especially if \texttt{\input} was redefined by some package. It was therefore changed to use \LaTeX{}\texttt{2e}'s internal name for this function. As a result, packages that modify \texttt{\end{document}} other than through the officially provided hooks may need to get updated.

Small improvement with split footnotes in \texttt{ftnright} If in the first column there is more than a full column worth of footnote material the material will be split resulting in footnotes out of order. This issue is now at least detected and generates an error but the algorithm used by the package is unable to gracefully handle it in an automated fashion (some alternatives for resolving the problem if it happens are given in the package documentation).

Improvement in \texttt{xspace} and font-switching The \texttt{xspace} package provides the command \texttt{xspace} which attempts to be clever about inserting spaces automatically after user-defined control sequences. An important bug fix has been made to this command to correct its behaviour when used in conjunction with font-switching commands such as \texttt{\emph} and \texttt{\textbf}. Previously, writing
\begin{verbatim}
\newcommand{\foo{foo}{xspace}
... \emph{\foo} bar baz
... \emph{\foo}, bar baz
\end{verbatim}
would result in an extraneous space being inserted after ‘foo’ in both cases; this has now been corrected.

RTL in \texttt{multicol} The 1.7 release of \texttt{multicol} adds support for languages that are typeset right-to-left. For those languages the order of the columns on the page also needs to be reversed—something that wasn’t possible in earlier releases.
\footnote{http://www.tug.org/TUGboat/tpub/27-2/tb87hagen-gyre.pdf}
The new feature is supported through the commands \RLmulticolcolumns (switching to right-to-left typesetting) and \LRmulticolcolumns (switching to left-to-right typesetting) the latter being the default.

**Improve French babel interaction with varioref**

Extracting and saving the page number turned out to be a source of subtle bugs. Initially it was done through an \edef with a bunch of \expandafter commands inside. This posed a problem if the page number itself contained code which needed protection (e.g., pr/4080) so this got changed in the last release to use \protected@edef. However, that in turn failed with Babel (bug report/4093) if the label contained active characters, e.g., a “:” in French. So now we use (after one failed attempt pr/4159) even more \expandafter commands and \romannumeral trickery to avoid any expansion other than what is absolutely required—making the code in that space absolutely unreadable.

\expandafter\def\expandafter#1\expandafter{\%
\romannumeral \expandafter\expandafter\expandafter \z@\expandafter@cdr\romannumeral \expandafter\expandafter\expandafter \z@\expandafter@cdr\csname r@#2\endcsname\@nil}%

Code like this nicely demonstrates the limitations in the programming layer of \LaTeX{} and the advantages that expl3 will offer on this level.
**Scheduled \LaTeX{} bug-fix release**

This issue of \LaTeX{} News marks the second bug-fix release of \LaTeX{} 2e (standard \LaTeX{}) since shifting to a new build system in 2009. Provided sufficient changes are made, we expect to make such releases yearly or every two years, in sync with \TeX{} Live.

**Release notes**

This release makes no changes to the core code in the \LaTeX{} 2e format but there are a small number of documentation fixes (not listed here). In addition several packages in the base and required areas have been updated as detailed below.

This has been done in accordance with the philosophy of minimising problems in both forwards and backwards compatibility, so most of these changes should not be noticed by the regular \LaTeX{} user.

References in the text below of the form “graphics/3873” are to bug reports listed at: http://latex-project.org/cgi-bin/ltxbugs2html

**fixltx2e updates**

There are a number of bugs and faulty design decisions in \LaTeX{} 2e that should have been corrected long ago in the kernel code. However, such corrections cannot be done as this would break backwards compatibility in the following sense. A large number of documents exist by now that have worked around the bug or have even made use of a particular misfeature. Thus changing the kernel code would break too many existing documents.

The corrections for these types of bug have therefore been collected together in a package that can be loaded only when needed; its name is fixltx2e. For this release we made the following changes to this package:

- Misspelled float placement specifiers such as `\begin{figure}[tv]` instead of `tb` are silently ignored by the kernel code. Now we test for such letters and issue an error message.

- \LaTeX{}’s float handling algorithm can get out of sync if you mix single and double-column floats (as they are placed independently of each other). This was corrected in fixltx2e a few years ago but the fix was not perfect as one situation using `\enlargethispage` generated a low-level \TeX{} error. This behaviour of the package is now improved.

**New fltrace package**

For years the file `1toutput.dtx` contained some hidden code to trace the detailed behaviour of the float placement algorithm of \LaTeX{}. Prompted by questions on StackExchange we now extract this code into a new `fltrace` package. To see the float algorithm in action (or to understand why it decides to place all your floats at the very end of the document) use `\usepackage{fltrace} \tracefloats`

To stop tracing somewhere in the document use `\tracefloatsoff` and to see the current value of various float parameters use `\tracefloatvals`. As the package is identical to the kernel code with tracing added, it may or may not work if you load any other package that manipulates that part of the kernel code. In such a case your best bet is to load `fltrace` first.

**inputenc package updates**

The `inputenc` package allows different input encodings for \LaTeX{} documents to be specified including the important `utf8` option used to specify the Unicode UTF-8 encoding. A common mistake in documents has been to also include this option when using the Unicode-based \TeX{} engines Lua\TeX{} and Xe\TeX{} producing strange errors as these engines natively deal with UTF-8 characters.

If a document stored in an 8bit encoding is processed by pdf\TeX{}, it needs the `inputenc` package to work correctly. However, if such a document is processed unchanged by Lua\TeX{} or Xe\TeX{}, then accented characters may silently get dropped from the output.

The package has been modified so that if used with Lua\TeX{} or Xe\TeX{}, then it just issues a warning if `utf8` or `ascii` is specified, and stops with an error for any other encoding requested.

One further improvement has been made to the encoding definition files (`.def`) used by `inputenc`: the catcode of `@` is now saved and restored when reading them instead of always using `\makeatother` inside the files (latex/4192).

**The tools directory**

In the past each of the sub-directories in the “required” section of the \LaTeX{} distribution contained a single `.ins` file to generate the code files from the source files. We have now started to provide individual `.ins` files for each of those packages that are likely to require updates outside a major \LaTeX{} release.
**multicol updates**

Version 1.8 of *multicol* implements some improvements/fixes and one extension. In the past the balancing algorithm enlarged the column height until it found a solution that satisfied all constraints. If there were insufficient break points then the final column height could have been much larger than expected and if that happened near the end of the page it resulted in the text overflowing into the bottom margin. This situation is now detected and in that case a normal page is cut and balancing is resumed on the next page. Some overflow is still allowed and controlled via the parameter `\maxbalancingoverflow`.

The use of `\enlargethispage` is now properly supported within the environment. Finally a new command `\docolaction` was added to allow the execution of code depending on the column in which the command is executed. See the documentation for details.

Bug fixes: the new version fixes both a color leak that could happen in certain situations and the problem that *multicols* could mess up the positioning of `\marginpars` that followed the environment.

**tabularx updates**

The restrictions on embedding `\tabularx` `\endtabularx` into the definition of a new environment have been relaxed slightly. See the package documentation for details.

**showkeys updates**

The `showkeys` package has been updated to fix problems if used at the start of list items, and to work if brace groups ({ and }) are used in the optional argument of `\cite`. (tools/4162, tools/4173)

**color updates**

The `\nopagecolor` command suggested by Heiko Oberdiek, available for some years in the `pdftex` option, has been added to the core package as suggested in graphics/3873. Currently this is supported in the driver files for `dvips` and `pdftex`. Patches to support other drivers are welcome.

**graphicx updates**

The `graphicx` version of `\rotatebox` now allows `\par` (and blank lines) in values, to match the change made to the `graphics` version some years ago. See graphics/4296.

**keyval updates**

All parsing used in the `keyval` package has been changed to allow `\par` (and blank lines) in values. (A second change, to parsing of brace groups in a construct such as `key={{{value}}}`, was reverted in v1.15.) See graphics/3446.

---

**Standard \LaTeX (\LaTeX2ε) and expl3**

The substantial collection of innovative code in *expl3* implements a new programming language that has for a while now been used by some writers of \LaTeX2ε packages. This code has recently also been made available for use on top of plain \TeX or Con\TeXt, largely to support generic packages that are supposed to work with different flavours of \TeX. These uses in no way affect authors of \LaTeX documents and such \LaTeX2ε packages will continue to work as advertised by their authors with standard \LaTeX.

This code base will also become an important foundation for the kernel of \LaTeX3 and so the new programming language can be described as ‘The \LaTeX3 Programming Language’. However, if you see or hear that a package ‘uses \LaTeX3’ then it remains very unlikely (as yet) to mean that the package is part of some ‘new version of \LaTeX’.

News about the development and use of *expl3* and about other developments in the \LaTeX3 code base is reported regularly in the \LaTeX3 News series ([http://latex-project.org/l3news/](http://latex-project.org/l3news/)), the most recent issue of which was published in March 2014.
New \LaTeX{} 2ε bug-fix policy

Introduction

For some years we have supplied bug fixes to the \LaTeX{} 2ε kernel via the fixltx2e package. This kept the kernel stable, but at the expense of meaning that most users did not benefit from bug fixes, and that some compromises which were made to save space in the machines of the time are still affecting most users today.

In this release we have started a new update policy. All the fixes previously available via fixltx2e are now enabled by default in the format, as are some further extensions for extended \TeX{} engines, \epsilon{}-\TeX{}, \Xe\TeX{} and \Lua\TeX{}. Compatibility and stability are still important considerations, and while most users will not notice these improvements, or will want to benefit from them, a new latexrelease package is provided that will revert all the changes and re-instate the definitions from earlier releases. The package can also be used with older releases to effectively update the kernel to be equivalent to this 2015 release.

A new document, \texttt{latexchanges}, is distributed with the release that documents all the changes to documented commands since the 2014 \LaTeX{} release, and will be updated in future releases if further changes have been made.

The latexrelease package

As noted above a new package is available to manage differences between \LaTeX{} releases. If you wish to revert all changes back to the definitions as they were in previous releases you may start your document requesting the \LaTeX{} release from May 2014:

\begin{verbatim}
\RequirePackage[2014/05/01]{latexrelease}
\documentclass{article}
\end{verbatim}

Conversely if you start a large project now and want to protect yourself against possible future changes, you may start your document requesting the \LaTeX{} release from January 2015:

\begin{verbatim}
\RequirePackage[2015/01/01]{latexrelease}
\documentclass{article}
\end{verbatim}

Then the version of latexrelease distributed with any future \LaTeX{} release will revert any changes made in that format, and revert to the definitions as they were at the beginning of 2015.

If you wish to share a document using the latest features with a user restricted to using an older format, you may use the form above and make the latexrelease package available on the older installation. The package will then update the format definitions as needed to enable the older format to work as if dated on the date specified in the package option.

The \texttt{\IncludeInRelease} command

The mechanism used in the latexrelease package is available for use in package code. If in your \texttt{zzz} package you have

\begin{verbatim}
\RequirePackage[2015/01/01]{latexrelease}
\IncludeInRelease{2015/06/01}{\zzz}{\zzz definition}
\def\zzz...new code
\EndIncludeInRelease
\INCLUDEINRELEASE{0000/00/00}{\zzz}{\zzz definition}
\def\zzz...original
\EndIncludeInRelease
\end{verbatim}

then in a document using a format dated 2015/06/01 or later, the “new code” will be used, and for documents being processed with an older format, the “original” code will be used. Note the format date here may be the original format date as shown at the start of every \LaTeX{} run, or a format date specified as a package option to the latexrelease package.

So if the document has

\begin{verbatim}
\RequirePackage[2014/05/01]{latexrelease}
\documentclass{article}
\usepackage{zzz}
\end{verbatim}

then it will use the original definition of \texttt{zzz} even if processed with the current format, as the format acts as if dated 2014/05/01.

Limitations of the approach

The new concept provides full backward and forward compatibility for the \LaTeX{} format, i.e., with the help of a current latexrelease package the kernel can emulate all released formats (starting with 2014/06/01\footnote{Patching an older format most likely works too, given that the changes in the past have been minimal, though this isn’t guaranteed and hasn’t been tested}).

However, this is not necessarily true for all packages. Only if a package makes use of the \texttt{\IncludeInRelease} functionality will it adjust to the requested \LaTeX{} release date. Initially this will only be true for a few selected packages and in general it may not even be
advisable for packages that have their own well-established release cycles and methods.

Thus, to regenerate a document with 100% compatible behavior it will still be necessary to archive it together with all its inputs, for example, by archiving the base distribution trees (and any modifications made). However, the fact that a document requests a specific \LaTeX{} release date should help identifying what release tree to use to achieve perfect accuracy.

Updates to the kernel

Updates incorporated from fixltx2e

The detailed list of changes incorporated from fixltx2e is available in the new \texttt{latexchanges} document that is distributed with this release. The main changes are that 2-column floats are kept in sequence with one column floats, corrections are made to the \texttt{\mark} system to ensure correct page headings in 2-column documents, several additional commands are made robust.

\texttt{\LaTeX} register allocation

\LaTeX{} has traditionally used allocation routines inherited from plain \TeX{} that allocated registers in the range 0–255. Almost all distributions have for some years used \texttt{\LaTeX} based formats (or \texttt{Xe\TeX} or Lua\TeX) which have 2\textsuperscript{15} registers of each type (2\textsuperscript{16} in the case of Lua\TeX{}). The \texttt{etex} package has been available to provide an allocation mechanism for these extended registers but now the format will by default allocate in a range suitable for the engine being used. The new allocation mechanism is different than the \texttt{etex} package mechanism, and supports Lua\TeX{}'s full range and an allocation mechanism for \texttt{I\LaTeX} floats as described below.

On \texttt{\LaTeX} based engines, an additional command, \texttt{\newmarka} is available (as with the \texttt{etex} package) that allocates extended \texttt{\LaTeX} marks, and similarly if \texttt{Xe\TeX} is detected a new command \texttt{\newXeTeXintercharclass} is available, this is similar to the command previously defined in the \texttt{xelatex.ini} file used to build the \texttt{xelatex} format.

Additional \LaTeX{} float storage

\LaTeX{}'s float placement algorithm needs to store floats (figures and tables) until it finds a suitable page to output them. It allocates 18 registers for this storage, but this can often be insufficient. The contributed \texttt{morefloats} package has been available to extend this list; however, it also only allocates from the standard range 0–255 so cannot take advantage of the extended registers. The new allocation mechanism in this release incorporates a new command \texttt{\extrafloats}. If you get the error: \texttt{Too many unprocessed floats}, then you can add (say) \texttt{\extrafloats{500}} to the document preamble to make many more boxes available to hold floats.

Built-in support for Unicode engines

The kernel sources now detect the engine being used and adjust definitions accordingly, this reduces the need for the \texttt{.ini} files used to make the formats to patch definitions defined in \texttt{latex.ltx}.

As noted above the format now includes extended allocation routines.

The distribution includes a file \texttt{unicode-letters.def} derived from the Unicode Consortium's Unicode Character Data files that details the upper and lower case transformation data for the full Unicode range. This is used to set the \texttt{1code} and \texttt{uccode} values if a Unicode engine is being used, rather than the values derived from the \texttt{T1} font encoding which are used with 8-bit engines.

Finally \texttt{\typein} is modified if Lua\TeX{} is detected such that it works with this engine.

\texttt{l3build}

This release has been tested and built using a new build system implemented in \texttt{Lua}, intended to be run on the \texttt{texlua} interpreter distributed with modern \TeX{} distributions. It is already separately available from CTAN. This replaces earlier build systems (based at various times on \texttt{make}, \texttt{cons}, and Windows \texttt{bat} files). It allows the sources to be tested and packaged on a range of platforms (within the team, OS X, Windows, Linux and Cygwin platforms are used). It also allows the format to be tested on \texttt{Xe\TeX} and Lua\TeX{} as well as the standard \texttt{pdf\TeX}/\texttt{\LaTeX} engines.

Hyperlinked documentation and TDS zip files

As well as updating the build system, the team have looked again at exactly what gets released to CTAN. Taking inspiration from Heiko Oberdiek's \texttt{latex-tds} bundle, the PDF documentation provided now includes hyperlinks where appropriate. This has been done without modifying the sources such that users without \texttt{hyperref} available can still typeset the documentation using only the core distribution. At the same time, the release now includes ready-to-install TDS-style zip files. This will be of principal interest to \TeX{} system maintainers, but end users with older machines who wish to manually update \LaTeX{} will also benefit.
Enhanced support for Lua\TeX

As noted in \LaTeX\ News 22, the 2015/01/01 release of \LaTeX\ introduced built-in support for extended \TeX\ systems.

The range of allocated register numbers (for example, for count registers) is now set according to the underlying engine capabilities to 256, 32768 or 65536. Additional allocators were also added for the facilities added by ε-\TeX\ (\texttt{newmark}) and Χ-\TeX\ (\texttt{newXeTeXintercharclass}). At that time, however, the work to incorporate additional allocators for Lua\TeX\ was not ready for distribution.

The main feature of this release is that by default it includes allocators for Lua\TeX\-provided features, such as Lua functions, bytecode registers, catcode tables and Lua callbacks. Previously these features have been provided by the contributed \texttt{luatex} (Heiko Oberdiek) and \texttt{luatexbase} (Élie Roux, Manuel Pégourié-Gonnard and Philipp Gesang) packages. However, just as noted with the \texttt{etex} package in the previous release, it is better if allocation is handled by the format to avoid problems with conflicts between different allocation schemes, or definitions made before a package-defined allocation scheme is enabled.

The facilities incorporated into the format with this release, and described below, are closely modelled on the \texttt{luatexbase} package and we thank the authors, and especially Élie Roux, for help in arranging this transition.

The implementation of these Lua\TeX\ features has been redesigned to match the allocation system introduced in the 2015/01/01 \LaTeX\ release, and there are some other differences from the previous \texttt{luatexbase} package. However, as noted below, \texttt{luatexbase} is being updated in line with this \LaTeX\ release to provide the previous interface as a wrapper around the new implementation, so we expect the majority of documents using \texttt{luatexbase} to work without change.

Names of Lua\TeX\ primitive commands

The 2015/01/01 \LaTeX\ release for the first time initialised Lua\TeX\ in \texttt{latex.ltx} if Lua\TeX\ is being used. Following the convention used in the contributed \texttt{luatex.ini} file used to set up the format for earlier releases, most Lua\TeX\-specific primitives were defined with names prefixed by \texttt{luatex}. This was designed to minimize name clashes but had the disadvantage that names did not match the Lua\TeX\ manual, or the names used in other formats, and produced some awkward command names such as \texttt{\luatexluafunction}. From this release the names are enabled without the \texttt{luatex} prefix.

In practice this change should not affect many documents; relatively few packages access the primitive commands, and many of those are already set up to work with prefixed or unprefixed names, so that they work with multiple formats.

For package writers, if you want to ensure that your code works with this and earlier releases, use prefixed names in the package and ensure that they are defined by using code such as:

\begin{verbatim}
\directlua{tex.enableprimitives("",
    tex.extraprimitives(
        "omega", "aleph", "luatex"))}
\end{verbatim}

Conversely if your document uses a package relying on prefixed names then you can add:

\begin{verbatim}
\directlua{tex.enableprimitives("luatex",
    tex.extraprimitives(
        "omega", "aleph", "luatex"))}
\end{verbatim}

to your document.

Note the compatibility layer offered by the \texttt{luatexbase} package described below makes several commands available under both names.
As always, this change can be reverted using: \texttt{\LaTeXVersion[2015/01/01]{latexrelease}} at the start of the document.

\texttt{\LaTeX\ commands for allocation in \LaTeXeX}

For detailed descriptions of the new allocation commands see the documented sources in \texttt{ltluatex.dtx} or chapter N of \texttt{source2e}; however, the following new allocation commands are defined by default in \LaTeXeX:\texttt{\newattribute, newcatcodetable, \newluafunction} and \texttt{\newwhatsit}. In addition, the commands \texttt{\setattribute} and \texttt{\unsetattribute} are defined to set and unset Lua attributes (integer values similar to counters, but attached to nodes). Finally several catcode tables are predefined:\texttt{\catcodetable@latex}, \texttt{\catcodetable@atletter}, \texttt{\catcodetable@string}, \texttt{\catcodetable@initex}, \texttt{\catcodetable@atletter}.

\textit{Predefined Lua functions}

If used with \texttt{\LuaTEX}, \texttt{\TeX} will initialise a Lua table, \texttt{luatexbase}, with functions supporting allocation and also the registering of Lua callback functions.

\textit{Support for older releases and plain \TeX}

The \texttt{\LuATEX} allocation functionality made available in this release is also available in plain \texttt{\TeX} and older \texttt{\LuATEX} releases in the files \texttt{ltluatex.tex} and \texttt{ltluatex.lua} which may be used simply by including the \texttt{\TeX} file: \texttt{\input{ltluatex}}. An alternative for old \texttt{\LuATEX} releases is to use: \texttt{\LaTeXVersion[2015/10/01]{latexrelease}} which will update the kernel to the current release, including \texttt{\LuATEX} support.

\textit{Additional \texttt{\LuATEX} support packages}

In addition to the base \texttt{\LuATEX} release two packages have been contributed to the contrib area on CTAN. The \texttt{ctablestack} package offers some commands to help package writers control the \texttt{\LuATEX catcodetable} functionality, and the \texttt{luatexbase} package replaces the previously available package of the same name, providing a compatible interface but implemented over the \texttt{ltluatex} code.

\textit{More Floats and Inserts}

If \texttt{\textcircled{S}-\TeX} is available, the number of registers allocated in the format to hold floats such as figures is increased from 18 to 52.

The extended allocation system introduced in 2015/01/01 means that in most cases it is no longer necessary to load the \texttt{etex} package. Many classes and packages that previously loaded this package no longer do so. Unfortunately in some circumstances where a package or class previously used the \texttt{etex} \texttt{\reserveinserts} command, it is possible for a document that previously worked to generate an error “no room for a new insert”. In practice this error can always be avoided by declaring inserts earlier, before the registers below 256 are all allocated. However, it is better not to require packages to be re-ordered and in some cases the re-ordering is complicated due to delayed allocations in \texttt{\AtBeginDocument}.

In this release, a new implementation of \texttt{\newinsert} is used which allocates inserts from the previously allocated float lists once the classical register allocation has run out. This allows an extra 52 (or in \texttt{\LuATEX}, 64 thousand) insert allocations which is more than enough for practical documents (by default, \texttt{\EuTEX} only uses two insert allocations).

\textit{Updated Unicode data}

The file \texttt{unicode-letters.def} recording catcodes, upper and lower case mappings and other properties for Unicode characters has been regenerated using the data files from Unicode 8.0.0.

\textit{Support for Comma Accent}

The command \texttt{\textcommabelow} has been added to the format. This is mainly used for the Romanian letters \texttt{sTt}. This was requested in latex/4414 in the \texttt{\EuTEX} bug tracker.

\textit{Extended inputenc}

The \texttt{utf8} option for \texttt{inputenc} has been extended to support the letters s and t with comma accent, U+0218–U+021b. Similarly circumflex w and y U+0174–U+0177 are defined. Also U+00a0 and U+00ad are declared by default, and defined to be \texttt{\nobreakspace} and \texttt{\textdagger} respectively.

The error message given on undefined UTF-8 input characters now displays the Unicode number in U+\texttt{hex} format in addition to showing the character.

\textit{Pre-release Releases}

The patch level mechanism has been used previously to identify \texttt{\textcircled{S}-\TeX} releases that have small patches applied to the main release, without changing the main format date.

The mechanism has now been extended to allow identification of pre-release versions of the software (which may or may not be released via CTAN) but can be identified with a banner such as \texttt{\LaTeXeXe \textcircled{S} \textcircled{S} \textcircled{S} 2015/10/01 pre-release-1}

Internally this is identified as a patch release with a negative patch level.

\textit{Updates in tools}

The \texttt{multicol} package has been updated to fix the interaction with “here” floats that land on the same page as the start or end of a \texttt{multicols} environment.
Contents

Lua\TeX\ support 34
Unicode data 34
More support for east European accents 35
Changes in Graphics 35
Changes in Tools 35
Improving support for Unicode engines 35

Lua\TeX\ support

This release refines the Lua\TeX\ support introduced in the 2015/10/01 release. A number of patches have been added to improve the behavior of \ltluatex (thanks largely to code review by Philipp Gesang). The kernel code has been adjusted to allow for changes in Lua\TeX\ v0.85–v0.88. Most notably, newer Lua\TeX\ releases allow more than 16 write streams and these are now enabled for use by \newwrite, but also the experimental \newtoken Lua library has been renamed back to token which required small adjustments in the \luaTeX setup.

The biggest change in Lua\TeX\ v0.85–v0.87 compared to previous versions is that all the primitives (originally defined in pdf\TeX) dealing with the PDF “back end” are no longer defined, being replaced by a much smaller set of new primitives. This does not directly affect the core \LaTeX files in this release but has required major changes to the .ini files used by \TeX Live and similar distributions to set up the format files. These changes in the Lua\TeX engine will affect any packages using these back end commands (packages such as graphics, color, hyperref, etc.). Until all contributed packages are updated to the new syntax users may need to add aliases for the old pdf\TeX commands. A new luapdftexalias package has been contributed to CTAN (not part of the core \LaTeX release) that may be used for this purpose.

See also the sections below for related changes in the tools and graphics bundles.

Unicode data

As noted in \LaTeX News 22, the 2015/01/01 release of \LaTeX introduced built-in support for extended \TeX systems. In particular, the kernel now loads appropriate data from the Unicode Consortium to set \lccode, \uccode, \catcode and \sfcode values in an automated fashion for the entire Unicode range.

The initial approach taken by the team was to incorporate the existing model used by (plain) X\TeX and to pre-process the “raw” Unicode data into a ready-to-use form as unicode-letters.def. However, the relationship between the Unicode Consortium files and \TeX data structures is non-trivial and still being explored. As such, it is preferable to directly parse the original (.txt) files at point of use. The team has therefore “spun-out” both the data and the loading to a new generic package, unicode-data. This package makes the original Unicode Consortium data files available in the \texmf tree (in tex/generic/unicode-data) and provides generic loaders suitable for reading this data into the plain, \LaTeX\ 2ε, and other, formats.

At present, the following data files are included in this new package:

- CaseFolding.txt
- EastAsianWidth.txt
- LineBreak.txt
- MathClass.txt
- SpecialCasing.txt
- UnicodeData.txt

These files are used either by \LaTeX\ 2ε or by expl3 (i.e. they represent the set currently required by the team). The Unicode Consortium provides various other data files and we would be happy to add these to the generic package, as it is intended to provide a single place to collect this material in the \texmf tree. Such requests can be mailed to the team as usual or logged at the package home page: https://github.com/latex3/unicode-data.

The new approach extends use of Unicode data in setting \TeX information in two ways. First, the \sfcode of all end-of-quotational/closing punctuation is now set to 0 (transparent to \TeX). Second, \umathcode values are now set using MathClass.txt rather than setting up only letters (which was done using an arbitrary plane 0/plane 1 separation). There are also minor refinements to the existing code setting, particularly splitting the concepts of case and letter/non-letter category codes.

For X\TeX, users should note that \xtxHanGlue and \xtxHanSpace are no longer defined, that no
assignments are made to \texttt{\XeTeXinterchartoks} and that no \texttt{\XeTeXinterchartoks} data is loaded into the format. The values which were previously inherited from the plain Xe\TeX\ setup files are not suitable for properly typesetting East Asian text. There are third-party packages addressing this area well, notably those in the CTeX bundle. Third-party packages may need adjustment to load the data themselves; see the unicode-data package for one possible loader.

**More support for east European accents**

As noted in \TeX\ News 23, comma accent support was added for \TeX{} and \LaTeX{} in the 2015/10/01 release. In this release a matching \texttt{\textcommaabove} accent has been added for U+0123 (\texttt{c{g}}, \texttt{g}) which is the lower case of U+0122 (\texttt{c{G}}, \texttt{G}). In the OT1 and T1 encodings the combinations are declared as composites with the \texttt{c} command, which matches the Unicode names “latin (capital|small) letter g with cedilla” and also allows \texttt{\MakeUppercase{c{g}}} to produce \texttt{c{G}}, as required. In T1 encoding, the composite of \texttt{c} with \texttt{k}, \texttt{n} and \texttt{r} are also declared to use the comma below accent rather than cedilla to match the conventional use of these letters.

The UTF-8 inputenc option utf8 has been extended to support all latin combinations that can be reasonably constructed with a (single) accent command an a base character for the T1 encoding so \texttt{g}, \texttt{u ˛} and similar characters may be directly input using UTF-8 encoding.

**Changes in Graphics**

The changes in \LaTeX{} v0.87 mean that the color and graphics packages no longer share the \texttt{pdftex} file between \LaTeX{} and \pdfTeX{}. A separate file \texttt{luatex.def} (distributed separately) has been produced, and distributions are encouraged to modify \texttt{graphics.cfg} and \texttt{color.cfg} configuration files to default to the \texttt{luatex} option if \LaTeX{} v0.87 or later is being used. The team has contributed suitable .cfg files to CTAN to be used as models.

Normally it is best to let the local \texttt{graphics.cfg} automatically supply the right option depending on the \TeX{} engine being used; however the color and graphics (and so graphics) packages have been extended to have an explicit \texttt{luatex} option comparable to the existing \texttt{pdftex} and \texttt{xetex} options.

The \texttt{trig} package has been updated so that pre-computed values such as \texttt{sin(90)} now expand to digits (1 rather than the internal token \texttt{\%one} in this case). This allows them to be used directly in PDF literal strings.

**Changes in Tools**

\LaTeX{} from version v0.87 no longer supports the \texttt{\write18} syntax to access system commands. A new package \texttt{shellesc} has been added to tools that defines a new command \texttt{\ShellEscape} that may be used in all \TeX{} variants to provide a consistent access to system commands. The package also defines \texttt{\write18} in Lua\TeX{} so that it continues to access system commands as before; see the package documentation for details.

**Improving support for Unicode engines**

Stability concerns are always paramount when considering any change to the \TeX{} 2E kernel. At the same time, it is important that the format remains usable and gives reliable results for users. For the Unicode \TeX{} engines Xe\TeX{} and Lua\TeX{} there are important differences in behavior from classical (8-bit) \TeX{} engines which mean that identical default behaviors are not appropriate. Over the past 18 months the team has addressed the most pressing of these considerations (as detailed above and in \TeX{} News 22 and 23), primarily by integrating existing patches into the kernel. There are, though, important areas which still need consideration, and which may result in refinements to kernel support in this area in future releases.

The default font setup in \TeX{} 2E at present is to use the OT1 encoding. This assumes that hyphenation patterns have been read using appropriate codes: the T1 encoding is assumed. The commonly-used hyphenation patterns today, \texttt{hyph-utf8}, are set up in this way for 8-bit engines (pdf\TeX{}) but for Unicode engines use Unicode code points. This means that hyphenation will be incorrect with Unicode engines unless a Unicode font is loaded. This requires a concept of a Unicode font encoding, which is currently provided by the \texttt{fontspec} package in two versions, EU1 and EU2. The team is working to fully understand what is meant by a “Unicode font encoding”, as unlike a classical \TeX{} encoding it is essentially impossible to know what glyphs will be provided (though each slot is always defined with the same meaning). There is also an overlap between this area and ideas of language and writing system, most obviously in documents featuring mixed scripts (for example Latin and Cyrillic).

As well as these font considerations, the team is also exploring to what extent it is possible to allow existing (8-bit) documents to compile directly with Unicode engines without requiring changes in the sources. Whether this is truly possible remains an open question.

It is important to stress that changes will only be made in this area where they do not affect documents processed with \texttt{\varepsilon-\TeX{}/pdf\TeX{}} (i.e. documents which are written for “classical” 8-bit \TeX{} engines). Changes will also be made only where they clearly address deficiencies in the current setup for Unicode engines (i.e. where current behaviors are wrong).
Lua\TeX

This \LaTeX\ release sees several internal changes designed to ensure that the system is still usable with Lua\TeX\ versions greater than 0.80, which have introduced many changes into the engine, most notably the removal or renaming of most of the primitive commands introduced by pdf\TeX. Also the lists of Lua callbacks handled by the callback allocation mechanism has been updated to match the callbacks defined in Lua\TeX\ version 0.90.

These changes have also required updates in tools and \amsmath as described below.

This is the first release of \LaTeX\ for which the test suite reports no failures when used with Lua\TeX.

Documentation checksums

The doc package has always provided two mechanisms that were mainly intended to guard against file truncation or corruption when files were commonly distributed by email through unreliable mail gateways: a Character Table of the ASCII character set could be inserted (and checked) and a “checksum” (count of the number of backslashes in the code sections) could be checked. These features are not really needed with modern distribution mechanisms and can be a distraction when reading the source code and so have been removed. The doc package has been updated so that if you use a \CheckSum command then, as before, the number is checked; however, if you omit the command then no error or warning is given.

Updates to inputenc

The UTF-8 support in inputenc has been further extended with support for non-breaking hyphens and more dashes.

Updates in Tools

The varioref package has been updated with improved documentation of multilingual support, and avoiding unnecessary warnings in some cases with \ref\text\faraway.

The tabularx package’s handling of \endtabularx in environment definitions has been fixed to again match its documentation.

The bm package has been updated as required by the changes to \mathchardef in Lua\TeX.

\amsmath

Since the launch of \LaTeX~2\epsilon in 1993, the \amsmath bundle has been part of the required packages in the core \LaTeX\ distribution, with bug reports handled by the \LaTeX bug database at \url{https://latex-project.org/bugs-upload.html}.

The \amsmath packages and the \amscls classes have been maintained by the American Mathematical Society.

With this release a new arrangement has been agreed between the American Mathematical Society and the \LaTeX\3 project. The \LaTeX\3 project will take over maintenance of the \amsmath bundle, with the American Mathematical Society retaining maintenance of \amscls.

The recommended installation of these files in the \TeX\ directory structure remains unchanged as \text/latex/amsmath and \text/latex/amscls respectively.

This release of \amsmath includes several updates so that \amsmath does not generate errors when math is used with Lua\TeX\ v0.87+, which has changes to \mathchardef that are incompatible with the previous version of \amsmath. It also improves \dots handling so that \long macros are correctly handled (for example, \dots \Rightarrow now uses centered dots), as well as commands expanding to character tokens (for example, \times \dots \times will use centered dots with \times defined as in the unicode-math package).

Related updates

In addition to the updates in the core \LaTeX\ release, some files in the CTAN “contrib” area have also been updated. Notably there have been further updates to the unicode-data files; also, the files required to build plain and \LaTeX\ formats have now been submitted to CTAN as tex-ini-files. The addition of a new luatex option for graphics-related packages (luatex-def on CTAN) has required updates to the configuration files to select a default option and these have similarly been uploaded to CTAN as graphics-cfg. (Previously these files were maintained directly in the \TeX\ Live repository, and were not available on CTAN.)


\textbf{LATeX News}

Issue 26, January 2017 (\LaTeX\ release 2017-01-01)

\section*{Contents}

- \LaTeX\ 37

- Default encodings in Xe\LaTeX\ and Lua\LaTeX\ 37

- \textbackslash showhyphens in Xe\LaTeX\ 38

- The fixltx2e package 38

- The latexbug package 38

- Updates to amsmath 38

- Updates to tools 38

- An addendum to the release changes in 2015: page breaks and vertical spacing 38

\section*{\LaTeX \textregistered \TEX \textregistered}

In \LaTeX\ News 16 (December 2003) the team announced

\begin{quote}
We expect that within the next two years, releases of \LaTeX\ will change modestly in order to run best under an extended \TeX\ engine that contains the $\LaTeX$ primitives, e.g., $\LaTeX$ or pdf\TeX.
\end{quote}

and also said

\begin{quote}
Although the current release does not require $\LaTeX$ features, we certainly recommend using an extended \TeX\ engine, especially if you need to debug macros.
\end{quote}

For many years the team have worked on the basis that users will have $\LaTeX$ available but had not revisited the above statements formally. As of the January 2017 release of \LaTeX\ 2e, $\LaTeX$ is required to build the format, and attempting to build a format without the extensions will fail.

Practically, modern \TeX\ distributions provide the extensions in all engines other than the “pure” Knuth \TeX, and indeed parts of the format-building process already require $\LaTeX$, most notably some of the UTF-8 hyphenation patterns. As such, there should be no noticeable effect on users of this change.

The team expect to make wider use of $\LaTeX$ within the kernel in future; details will be announced where they impact on end users in a visible way.

\textbf{Default encodings in Xe\LaTeX\ and Lua\LaTeX}

The default encoding in \LaTeX\ has always been the original 128-character encoding OT1. For Unicode based \TeX\ engines, this is not really suitable, and is especially problematic with Xe\LaTeX\ as in the major distributions this is built with Unicode based hyphenation patterns in the format. In practice this has not been a major problem as documents use the contributed \texttt{fontspec} package in order to switch to a Unicode encoded font.

In this release we are adding TU as a new supported encoding in addition to the previously supported encodings such as OT1 and T1. This denotes a Unicode based font encoding. It is essentially the same as the TU encoding that has been on trial with the experimental tuenc option to \texttt{fontspec} for the past year.

The Xe\LaTeX\ and Lua\LaTeX\ formats will now default to TU encoding and \texttt{lmr} (Latin Modern) family. In the case of Lua\LaTeX\ the contributed \texttt{luaotfload} Lua module will be loaded at the start of each run to enable the loading of OpenType fonts.

The \texttt{fontspec} package is being adjusted in a companion release to recognise the new encoding default arrangements.

Note that in practice no font supports the full Unicode range, and so TU encoded fonts, unlike fonts specified for T1, may be expected to be incomplete in various ways. In the current release the file \texttt{tuenc.def} that implements the TU encoding-specific commands has made some basic assumptions for (for example) default handling of accent commands, and the set of command names is derived from the command names used for the UTF-8 support in the \texttt{inputenc} package, restricted roughly to the character ranges classically provided by the T1 and TS1 encodings, but is part of a longer term plan seen over recent releases to increase support for Unicode based \TeX\ engines into the core \LaTeX\ support.

If for any reason you need to process a document with the previous default OT1 encoding, you may switch encoding in the usual ways, for example

\begin{verbatim}
\usepackage[OT1]{fontenc}
\end{verbatim}

or you may roll back all the changes for this release by starting the document with

\begin{verbatim}
\RequirePackage[2016/12/31]{latexrelease}
\end{verbatim}
\texttt{showhyphens\ in \LaTeX\ \textsc{X}}

Due to the way \LaTeX\ \textsc{X} interfaces to font libraries, the standard definition of \texttt{showhyphens} does not work. A variant definition has been available in the contributed \texttt{xltt} package, however a (slightly different) definition for \texttt{showhyphens} is now included in \LaTeX\ \textsc{X} by default. As usual this change will be undone if an earlier release is specified using the \texttt{latexrelease} package.

\textbf{The \texttt{fixltt} package}

As described in \LaTeX\ News 22, the \texttt{fixltt} package has become obsolete with the new update policy. Since 2015 it has just made a warning and exited. In this release we have re-introduced all the code from the original fixes in the 2014 \LaTeX\ but guarded by \texttt{\UsePackage{fixltt}}. So for current releases \texttt{fixltt} still just displays a warning but for old releases, whether that is an old format, or a format with the version date reset via the \texttt{latexrelease} package, the fixes in the original \texttt{fixltt} will be applied.

This improves the ability to run old documents in a way that is compatible with contemporary formats. If you have a 2014 document that used \texttt{\UsePackage{fixltt}} and you add \texttt{\RequirePackage[\texttt{2014/01/01}]{latexrelease}} and process it with the current format then \texttt{latexrelease} will undo most changes made since 2014, but now when the document includes \texttt{fixltt} it will act like a 2014 version of the package and apply the code fixes, not just give a warning that the package is obsolete.

\textbf{The \texttt{latexbug} package}

As explained in more detail at the \LaTeX\ Project website\footnote{\url{https://www.latex-project.org/bugs/}} a new package, \texttt{latexbug}, has been produced to help produce test files to accompany bug reports on the core \LaTeX\ distribution. This is being published separately to CTAN at the same time as this release. By using the \texttt{latexbug} package you can easily check that the packages involved in the test are all part of the core release. The \LaTeX\ project cannot handle bug reports on contributed packages, which should be directed to the package maintainer as given in the package documentation.

\textbf{Updates to \texttt{amsmath}}

The \texttt{amsmath} package has two updates at this release.

- The large delimiters around generalised fractions (for example in the $\set\fixltx2e\binom$ construct) did not work in previous releases if using \TeX\ or \textsc{Xe\TeX} with OpenType math fonts. This is related to the lack of specific metrics for this use in the OpenType Math table. In principle \TeX\ has two additional named metrics to control the delimiters but these are not initialised by default, and in \textsc{Xe\TeX} it does not seem possible to make them work at all. So for Unicode \TeX\ systems, a new implementation of $\set\fixltx2e\genfrac$ is used at this release that uses $\set\fixltx2e\left\right$ internally but parameterised to give spacing as close to the original as possible. The implementation in (pdf)\TeX\ is unaffected.

\textbf{Updates to tools}

The \texttt{array} package has been updated to fix a longstanding but previously unreported issue with unwanted interactions between tables in the page head or foot and the body of the page, as reported in \texttt{PR tools/4188}. There is also an update to the \texttt{Lua\TeX} support in \texttt{bm}.

\textbf{An addendum to the release changes in 2015: page breaks and vertical spacing}

In 2015 we announced the introduction of the roll-back/roll-forward concept to manage bug fixes and additions to core \LaTeX\ in a manageable way. We also announced at that time that we now incorporate all fixes from \texttt{fixltt} into the kernel (as the old mechanism produced problems instead of improving the situation). Refer to \texttt{ltnews22.pdf} for details.

One of the fixes from \texttt{fixltt} was for a glaring bug in \texttt{addvspace} that was originally detected in the mid-nineties and back then added to the \texttt{fixltt} support package. In certain situations \texttt{addvspace} would result in a page/column break below the baseline of the last line. As a result documents using \texttt{\flushbottom} would show a clear misalignment (even more prominent when typesetting in two-column mode).

Starting with release 2015/01/01 this is now finally corrected already in the kernel and not only in \texttt{fixltt}.

In nearly all circumstances this will either make no difference to existing documents, or it will locally improve the visual appearance of that document without changing anything on other pages. However, by the nature of the change it is also possible that there are further non-local changes to the page breaks due to the different break positions introduced by the fix.

Thus, for documents that have been written before 2015 and that should be preserved unchanged at all costs you may have to add

\texttt{\RequirePackage[\texttt{2014/01/01}]{latexrelease}}

at the top of the document, to roll back the format to a date before the policy change.
ISO 8601 Date format

Since before the first releases of \LaTeX2ε, \LaTeX\ has used a date format in the form \texttt{YYYY/MM/DD}. This has many advantages over more conventional formats, as it is easy to sort and avoids the unfortunate ambiguity between different communities as to whether 01/02/2017 is the 1st of February or 2nd of January.

However there is another date format, formalised by the International Standard ISO 8601. The basic format defined by this standard is functionally equivalent to the \LaTeX\ format, but using \texttt{-} rather than \texttt{/}. This date format is now supported in many Operating Systems and applications (for example the \texttt{date \--iso-8601} command in Linux and similar systems).

From this release, \LaTeX\ will accept ISO format date strings in the date argument of \texttt{\ProvidesPackage}, \texttt{\usepackage}, etc. Currently we recommend that you do not use this format in any packages that need to work with older \LaTeX\ releases; the \texttt{latexrelease} package may be used with older releases to add this functionality. This change is handled in a special way by \texttt{latexrelease}: the package always adds support for ISO dates whatever format date is requested; this is required so that the necessary date comparisons may be made.

The new functionality can be seen in the startup banner which advertises \texttt{LaTeX2e <2017-04-15>}.

Further TU encoding improvements

The 2017/01/01 release saw the introduction of the new TU encoding for specifying Unicode fonts with Lua\TeX\ and \texttt{Xe\TeX}. There were a number of small corrections and additions in the patch releases updating 2017/01/01, and a further addition in this release, notably extended support for the dot-under accent, \texttt{\d}.

Disabling hyphenation

The existing \LaTeX\ code for \texttt{\verb} and \texttt{verbatim} had some issues when used with fonts that were not loaded with hyphenation disabled via setting \texttt{hyphenchar} to \texttt{-1}. In this release these verbatim environments use a \texttt{\language} setting, \texttt{\@nohyphenation}, that has no hyphenation patterns associated.

The format ensures that a language has been allocated with this name. For most users this will in fact be no change as the standard \texttt{babel} language has for a long time allocated a language with this name.

In order that page breaks in \texttt{verbatim} do not influence the language used in the page head and foot, the format now normalises the language used in the output routine to a default language as described below.

Discretionary hyphenation

The \LaTeX\ definition of \texttt{-} has been adjusted so that it will insert the current font’s \texttt{hyphenchar}, as would the \TeX\ primitive. A comment in \texttt{source2e} has given this new definition since the first releases of \LaTeX2ε, and in this release we finally acted upon this comment. Previously \texttt{-} always inserted a \texttt{-} at a break point even if a different character would be used for automatic hyphenation with the current font.

Default document language

A new integer parameter \texttt{\document@default@language} is introduced; this is initialised to \texttt{-1} but is set at \texttt{\begin{document}} to the language in force at that time if it has not been set by preamble code. This is very similar to the handling of the default color, and is used in a similar way to normalise the settings for page head and foot as described above. Users should not normally need to set this explicitly but it is expected that language packages such as \texttt{babel} may set this if the default behaviour is not suitable.

Line spacing in parboxes

Inside a \texttt{parbox} \LaTeX\ normalises the baseline spacing. However it has not previously reset \texttt{\lineskiplimit}. This meant that lines of a paragraph that have ascenders or descenders could be set with \texttt{closer} line spacing than lines without. This can easily happen if you use a \texttt{parbox} in an AMS alignment, as they use a relatively large value of \texttt{\lineskiplimit}. As usual, the \texttt{latexrelease} package may be used to force the older behavior.
**Contents**

A new home for \LaTeX\ 2\epsilon sources 40

Bug reports for core \LaTeX\ 2\epsilon 40

UTF-8: the new default input encoding 40

The new default ................................. 41
Compatibility ...................................... 41
BOM: byte order mark handling .................. 41

A general rollback concept 41

Integration of \remreset and \chngcntr packages 42

Testing for undefined commands 42

Changes to packages in the tools category 42

\LaTeX\ table columns with fixed widths ........ 42
Obscure overprinting with \multicol fixed ........ 42

Changes to packages in the amsmath category 42

Updated user’s guide ............................. 42

A new home for \LaTeX\ 2\epsilon sources

In the past the development version of the \LaTeX\ 2\epsilon source files has been managed in a Subversion source control system with read access for the public. This way it was possible to download in an emergency the latest version even before it was released to CTAN and made its way into the various distributions.

We have recently changed this setup and now manage the sources using Git and placed the master sources on GitHub at

https://github.com/latex3/latex2e

where we already store the sources for expl3 and other work. As before, direct write access is restricted to \LaTeX\ Project Team members, but everything is publicly accessible including the ability to download, clone (using Git) or checkout (using SVN). More details are given in [1].

Bug reports for core \LaTeX\ 2\epsilon

For more than two decades we used GNATS, an open source bug tracking system developed by the FSF. While that has served us well in the past it started to show its age more and more. So as part of this move we also decided to finally retire the old \LaTeX\ bug database and replace it with the standard “Issue Tracker” available at Github.

The requirements and the workflow for reporting a bug in the core \LaTeX\ software is documented at

https://www.latex-project.org/bugs/

and with further details also discussed in [1].

UTF-8: the new default input encoding

The first TEX implementations only supported reading 7-bit ASCII files—any accented or otherwise “special” character had to be entered using commands, if it could be represented at all. For example to obtain an “ä” one would enter \textbackslash a, and to typeset a “ß” the command \textbackslash ss. Furthermore fonts at that time had 128 glyphs inside, holding the ASCII characters, some accents to build composite glyphs from a letter and an accent, and a few special symbols such as parentheses, etc.

With 8-bit TEX engines such as pdfTEX this situation changed somewhat: it was now possible to process 8-bit files, i.e., files that could encode 256 different characters. However, 256 is still a fairly small number and with this limitation it is only possible to encode a few languages and for other languages one would need to change the encoding (i.e., interpret the character positions 0–255 in a different way). The first code points 0–127 were essentially normed (corresponding to ASCII) while the second half 128–255 would vary by holding different accented characters to support a certain set of languages.

Each computer used one of these encodings when storing or interpreting files and as long as two computers used the same encoding it was (easily) possible to exchange files between them and have them interpreted and processed correctly.

But different computers may have used different encodings and given that a computer file is simply a sequence of bytes with no indication for which encoding is intended, chaos could easily happen and has happened. For example, the German word “Größe” (height) entered on a German keyboard could show up as “GrŤàe” on a different computer using a different encoding by default.

So in summary the situation wasn’t at all good and it was clear in the early nineties that \LaTeX\ 2\epsilon (that was being developed to provide a \LaTeX\ version usable across the world) had to provide a solution to this issue.

The \LaTeX\ 2\epsilon answer was the introduction of the \inputenc package [2] through which it is possible to
provide support for multiple encodings. It also allows to correctly process a file written in one encoding on a computer using a different encoding and even supports documents where the encoding changes midway.

Since the first release of \LaTeX{} 2ε in 1994, \LaTeX{} documents that used any characters outside ASCII in the source (i.e. any characters in the range of 128–255) were supposed to load \inputenc and specify in which file encoding they were written and stored. If the \inputenc package was not loaded then \LaTeX{} used a “raw” encoding which essentially took each byte from the input file and typeset the glyph that happened to be in that position in the current font—something that sometimes produces the right result but often enough will not.

In 1992 Ken Thompson and Rob Pike developed the UTF-8 encoding scheme which enables the encoding of all Unicode characters within 8-bit sequences. Over time this encoding has gradually taken over the world, replacing the legacy 8-bit encodings used before. These days all major computer operating systems use UTF-8 to store their files and it requires some effort to explicitly store files in one of the legacy encodings.

As a result, whenever \LaTeX{} users want to use any accented characters from their keyboard (instead of resorting to "α and the like) they always have to use

\begin{verbatim}
\usepackage[utf8]{inputenc}
\end{verbatim}

in the preamble of their documents as otherwise \LaTeX{} will produce gibberish.

**The new default**

With this release, the default encoding for \LaTeX{} files has been changed from the “fall through raw” encoding to UTF-8 if used with classic \TeX{} or pdf\TeX{}. The implementation is essentially the same as the existing UTF-8 support from \usepackage\{utf8\}\{inputenc\}.

The Lua\TeX{} and Xe\TeX{} engines always supported the UTF-8 encoding as their native (and only) input encoding, so with these engines \inputenc was always a no-op.

This means that with new documents one can assume UTF-8 input and it is no longer required to always specify \usepackage\{utf8\}\{inputenc\}. But if this line is present it will not hurt either.

**Compatibility**

For most existing documents this change will be transparent:

- documents using only ASCII in the input file and accessing accented characters via commands;
- documents that specified the encoding of their file via an option to the \inputenc package and then used 8-bit characters in that encoding;
- documents that already had been stored in UTF-8 (whether or not specifying this via \inputenc).

Only documents that have been stored in a legacy encoding and used accented letters from the keyboard without loading \inputenc (relying on the similarities between the input used and the T1 font encoding) are affected.

These documents will now generate an error that they contain invalid UTF-8 sequences. However, such documents may be easily processed by adding the new command `\UseRawInputEncoding` as the first line of the file. This will re-instantiate the previous “raw” encoding default.

`\UseRawInputEncoding` may also be used on the command line to process existing files without requiring the file to be edited

```bash
pdf\LaTeX{} `\UseRawInputEncoding \input{file}` file
```

will process the file using the previous default encoding.

Possible alternatives are reencoding the file to UTF-8 using a tool (such as recode or iconv or an editor) or adding the line

```latex
\usepackage\{encoding\}\{inputenc\}
```

to the preamble specifying the \texttt{(encoding)} that fits the file encoding. In many cases this will be \texttt{latin1} or \texttt{cp1252}. For other encoding names and their meaning see the \inputenc documentation.

As usual, this change may also be reverted via the more general \latexrelease package mechanism, by specifying a release date earlier than this release.

**BOM: byte order mark handling**

When using Unicode the first bytes of a file may be a, so called, BOM character (byte order mark) to indicate the byte order used in the file. While this is not required with UTF-8 encoded files (where the byte order is known) it is nevertheless allowed by the standard and some editors add that byte sequence to the beginning of a file. In the past such files would have generated a “Missing begin document” error or displayed strange characters when loaded at a later stage.

With the addition of UTF-8 support to the kernel it is now possible to identify and ignore such BOMs characters even before `\documentclass` so that these issues will no longer be showing up.

**A general rollback concept for packages and classes**

In 2015 a rollback concept for the \LaTeX{} kernel was introduced. Providing this feature allowed us to make corrections to the software (which more or less didn’t happen for nearly two decades) while continuing to maintain backward compatibility to the highest degree.
In this release we have now extended this concept to the world of packages and classes which was not covered initially. As the classes and the extension packages have different requirements compared to the kernel, the approach is different (and simplified). This should make it easy for package developers to apply it to their packages and authors to use when necessary.

The documentation of this new feature is given in an article submitted to TUGboat and also available from our website [3].

Integration of remreset and chngcntr packages into the kernel

With the optional argument to \newcounter \LaTeX offers to automatically reset counters when some counter is stepped, e.g., stepping a chapter counter resets the section counter (and recursively all other heading counters). However, what was until now missing was a way to undo such a link between counters or to link two counters after they have been defined.

This can be now be done with \counterwithin and \counterwithout, respectively. In the past one had to load the chngcntr package for this. For the programming level we also added \@removefromreset as the counterpart of the already existing \@addtoreset command. Up to now this was offered by the remreset package.

Testing for undefined commands

\LaTeX packages often use a test \@ifundefined to test if a command is defined. Unfortunately this had the side effect of defining the command to \relax in the case that it had no definition. The new release uses a modified definition (using extra testing possibilities available in \TeX). The new definition is more natural, however code that was relying on the side effect of the command being tested being defined if it was previously undefined may have to add \let \langle \text{command} \rangle \relax.

Changes to packages in the tools category

\LaTeX table columns with fixed widths

Frank published a short paper in TUGboat [4] on producing tables that have columns with fixed widths. The outlined approach using column specifiers “\texttt{w}” and “\texttt{W}” has now been integrated into the array package.

Obscure overprinting with multicol fixed

A rather peculiar bug was reported on StackExchange for multicol. If the column/page breaking was fully controlled by the user (through \columnbreak) instead of letting the environment do its job and if then more \columnbreak commands showed up on the last page then the balancing algorithm was thrown off track. As a result some parts of the columns did overprint each other.

The fix required a redesign of the output routines used by multicol and while it “should” be transparent in other cases (and all tests in the regression test suite came out fine) there is the off-chance that code that hooked into internals of multicol needs adjustment.

Changes to packages in the amsmath category

With this release of \TeX a few minor issues with amsmath have been corrected.

Updated user’s guide

Furthermore, amstoc.pdf, the AMS user’s guide for the amsmath package [5], has been updated from version 2.0 to 2.1 to incorporate changes and corrections made between 2016 and 2018.

References


Introduction
The December 2018 release of \LaTeX{} is a maintenance release in which we have fixed a few bugs in the software: some are old, some newer, and they are mostly rather obscure.

Bug reports for core \LaTeX{} 2\epsilon{} and packages maintained by the Project Team
In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.

Changes to the \LaTeX{} kernel

UTF-8: updates to the default input encoding
In the April 2018 release of \LaTeX{} we changed the default encoding from 7-bit ascii to utf-8 when using classic \TeX{} or pdf\TeX{}; see \LaTeX{} News 28 [2] for details.

Now, after half a year of experience with this new default, we have made a small number of adjustments to further improve the user experience. These include:

- Some improvements when displaying error messages about utf-8 characters that have not been set up for use with \LaTeX{}, or are invalid for some other reason; (github issues 60, 62 and 63)
- The addition of a number of previously missing declarations for characters that are in fact available with the default fonts, e.g., \verb|"j| (0237), \verb|SS| (1E9E), \verb|˛| (02DB) and \verb|˙| (02D9);
- Correcting the names for \verb|«| and \verb|»| in all encoding files. These correct names are in addition to the old (but wrong) Adobe names: Adobe mistakenly called them Guillemot, which is a sea bird.

Fixed \verb*{} and friends in \TeX{} and Lua\TeX{}
The original \verb*{} and \verbatim*{} in \LaTeX{} were coded under the assumption that the position of the space character (i.e., ASCII 32) in a typewriter font contains a visible space glyph \verb\text{"}\text{␣.}\text{"}. This is correct for pdf\TeX{} with the most used font encodings OT1 and T1. However, this unfortunately does not work for Unicode engines using the \text{T}\text{U} encoding since the space character slot (ASCII 32) then usually contains a real (normal) space, which

Initial experience with this system is good, with people who report problems following the guidelines and including helpful working examples to show the problem—thanks for doing this.

The detailed requirements and the workflow for reporting a bug in the core \LaTeX{} software is documented at

https://www.latex-project.org/bugs/

with further details and discussion in [1].

Changes to the \LaTeX{} kernel

UTF-8: updates to the default input encoding
In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.

In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.

In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.

In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.

In Spring 2018 we established a new issue tracking system (Github issues at https://github.com/latex3/latex2e/issues) for both the \LaTeX{} core and the packages maintained by the \LaTeX{} Project Team, with an updated procedure for how to report a bug or problem.
has the effect that \verb* produces the same results as \verb.

The \verb* code now always uses the newly introduced command \verbvisiblespace to produce the visible space character and this command will get appropriate definitions for use with the different engines. With pdf\TeX it will simply use \asciispace, which is a posh name for “select character 32 in the current font”, but with Unicode engines the default definition is

\begin{verbatim}
\DeclareRobustCommand\verbvisiblespace
{\leavevmode
 \{\usefont{OT1}{cmtt}{m}{n}\asciispace\}}
\end{verbatim}

which uses the visible space from the font Computer Modern Typewriter, regardless of the currently chosen typewriter font. Internally the code ensures that the character used has exactly the same width as the other characters in the current (monospaced) font; thus, for example, code displays line up properly.

It is possible to redefine this command to select your own character, for example

\begin{verbatim}
\DeclareRobustCommand\verbvisiblespace
{\textvisiblespace}
\end{verbatim}

will select the “official” visible space character of the current font. This may look like the natural default, but it wasn’t chosen as our default because many fonts just don’t have that Unicode character, or they have one with a strange shape. \text{(github issues 69 and 70)\par}

\textbf{Error message corrected}

Trying to redefine an undefined command could in a few cases generate an error message with a missing space, e.g., \texttt{\renewcommand\1{\ldots}} gave

La\TeX Error: \1undefined.\par

This is now fixed. \text{(github issue 41)\par}

\textbf{Fixed fatal link error with hyperref}

If an \href link text gets broken across pages, pdf\TeX and Lua\TeX will generate a fatal error unless both parts of the link are internally at the same boxing level. In two-column mode that was not the case if one of the pages had spanning top floats. This has now been changed so that the error is avoided. \text{(github issue 94)\par}

\textbf{Avoid page breaks caused by invisible commands}

Commands like \texttt{\label} or \texttt{\index} could generate a potential page break in places where a page break was otherwise prohibited, e.g., when used between two consecutive headings. This has now been corrected. If for some reason you really want a break and you relied on this faulty behavior, you can always add one using \texttt{\pagebreak}, with or without an optional argument. \text{(github issue 81)\par}

\textbf{Prevent spurious spaces when reading table of contents data}

When table of contents data is read in from a .toc file, the new-line character at the end of each line is converted by \TeX to a space. In normal processing this is harmless (as \TeX is doing this input reading whilst in vertical mode and each line in the file represents a single line (paragraph) in the table of contents). If, however, this is done in horizontal mode, which is sometimes the case, then these spaces will appear in the output. If you then omit some of the input lines (e.g., because you do not display TOC data below a certain level), then these spaces accumulate in the typeset output and you get surprising, and unwanted, gaps inside the text.

The new code now adds a % sign at the end of problematic lines in the .toc file so that \TeX will not generate such spaces that may survive to spoil the printed result. As some third party packages have augmented or changed the core \TeX functionality in that area (for example, by adding additional arguments to the commands in TOC files) the code uses a conservative approach and the % signs are added only when certain conditions are met. Therefore some packages might require updates if they want to benefit from this correction, especially if they unconditionally overwrite \TeX’s \texttt{\addcontentsline} definition. \text{(github issue 73)\par}

\textbf{Prevent protrusion in table of contents lines}

In \TeX’s internal processing model, paragraph data is one of the major data structures. As a result, many things are internally modeled as paragraphs even if they are not conceptually “text paragraphs” in the traditional sense. In a few cases this has some surprising effects that are not always for the better. One example is standard TOC entries, where you have heading data followed by some dot leaders and a page number at the right, produced, for example, from this:

\verb|
\addcontentsline{toc}{section}{
\begin{itemize}
\item \texttt{\index} \texttt{label}
\item \texttt{\index} \texttt{label}
\end{itemize}
|
\verb|
\end{verbatim}

\textbf{Error message corrected} . . . . . . . . . . 2

The space reserved for the page number is of a fixed width, so that the dots always end in the same place. Well, they did end in the same place until the advent of protrusion support in the \TeX engines. Now, with the microtype package loaded, it is possible that the page number will protrude slightly into the margin (even though it’s typeset inside a box) and as a result this page number box gets shifted. With enough bad luck this can get you another dot in the line, sticking out like the proverbial sore thumb, as exhibited in the question on StackExchange that triggered the correction.

\TeX now takes care that there will be no protrusion happening on such lines, even if it is generally enabled for the whole document. \text{(https://tex.stackexchange.com/q/172785)\par}
Start L-R mode for \thinspace and friends
In \LaTeX, commands that are intended only for paragraph (L-R) mode are generally careful to start paragraph mode if necessary; thus they can be used at the start of a paragraph without surprising and unwanted consequences. This important requirement had been overlooked for a few horizontal spacing commands, such as \thinspace (a.k.a. "\,", and for some other support commands such as \smash or \phantom. Thus they ended up adding vertical space when used at the beginning of a paragraph or, in the case of \smash, creating a paragraph of their own. This has now been corrected, and a corresponding update has been made to the amsmath package, in which these commands are also defined.  

\textit{Guarding \pfill in doc}

For presenting index entries pointing to code fragments and the like, the doc package has a \pfill command that generates within the index a line of dots leading from the command name to the page or code line numbers. If necessary it would automatically split the entry over two lines. That worked well enough for a quarter century, but we discovered recently that it is broken inside the \texttt{ltugboat} class, where it sometimes produces bad spacing within continuation lines.

The reason turned out to be a redefinition of the \LaTeX\ command \nobreakspace (-) inside the class \texttt{ltugboat}, which removed any preceding space (and thus unfortunately also removed the dots on the continuation line). While one can argue that this is a questionable redefinition (if only done by a single class and not generally), it has been in the class so long that changing it would certainly break older documents. So instead we now guard against that removal of space.

Changes to packages in the tools category

\textit{Sometimes the trace package turned off too much}

The trace package is a useful little tool for tracing macro execution: it hides certain lengthy and typically uninteresting expansions resulting from font changes and similar activities. However, it had the problem that it also reset other tracing settings such as \showoutput in such situations, so that you couldn’t use \showoutput in the preamble to get symbolic output of all the pages in the document. This has now been corrected.

\textit{Update to xr}

The xr package has been updated so that the code that reads the .aux file has been made more robust. It now correctly ignores conditionals (added by \hyperref and other packages) rather than generating low level parsing errors.

\textit{Column data for multicols sometimes vanished}

In certain situations involving \texttt{multicols*}, when there are more explicit \columnbreak requests than there are columns on the current page, data could vanish due to the removal of an internal penalty marking the end of the environment. This has been corrected by explicitly reinserting that penalty if necessary.

\textit{Extension to \docolaction in multicol}

The \texttt{docolaction} command can be used to carry out actions depending on the column you are currently in, i.e., first, any inner one (if more than two) or last. However, if the action generates text then there is the question: is this text part of the current column or the one after? That is, on the next run, do we test before or after it, to determine in which column we are?

This is now resolved as follows: if you use \texttt{docolaction*} any generated text by the chosen action is considered to be after the test point. But if you use the command without the star then all the material it generates will be placed before the test point to determine the current column, i.e., the text will become part of the current column and may affect the test result on the next run.

\textit{Prevent color leak in array}

In some cases the color used inside a \texttt{tabular} cell could “leak out” into the surrounding text. This has been corrected.

\textit{Support fragile commands in array or tabular column templates}

The preamble specifiers \texttt{p, m and b} each receives a user supplied argument: the width of the paragraph column. Normally that is something harmless, like a length or a simple length expression. But in more complicated settings involving the \texttt{calc} package it could break with a low-level error message. This has now been corrected.

\textit{Changes to packages in the amsmath category}

The changes in the kernel made for \texttt{\thinspace, \smash}, etc. (see above) have been reflected in the amsmath package code, so that loading this package doesn’t revert them.

\textit{Website updates}

\textit{Publications area reorganized and extended}

To help readers to find relevant information in more convenient and easy ways, the area of the website covering publications by the \LaTeX\ Project Team was reorganized and extended (many more abstracts added).

We now provide the articles, talks and supplementary data structured both by year and also by major topics [4]. Feel free to take a look.
Japanese translations of the user’s guide
Yukitoshi Fujimura has kindly translated into Japanese
two documents that are distributed with standard
\LaTeX. These are:

- \LaTeX{}2ε for authors;
- User’s Guide for the \texttt{amsmath} Package [5].

They can be found on the website documentation
page [3]. You will now also find there a typeset version
of the full \LaTeX{}2ε source code (with index etc.) and a
number of other goodies.

References
the \LaTeX{} core software}. In: TUGboat, 39#1, 2018.
https://latex-project.org/publications/
https://latex-project.org/news/latex2e-news/
[3] \LaTeX{} documentation on the \LaTeX{} Project Website.
https://latex-project.org/documentation/
[4] \LaTeX{} Project publications on the \LaTeX{} Project
Website.
https://latex-project.org/publications/
[5] American Mathematical Society and The \LaTeX{} Project:
\textit{User’s Guide for the \texttt{amsmath} Package}
(Version 2.1). April 2018. Available from
https://www.ctan.org and distributed as part of
every \LaTeX{} distribution.
LATEX News
Issue 30, October 2019 (LATEX release 2019-10-01)

Contents

LATEX-dev formats now available 47
Our hopes 47
Details please 48
Setting up menu items 48

Improving Unicode handling in pdfTeX 48
Improving file name handling in pdfTeX 48
Improving the filecontents environment 48
Making more user commands robust 48

Other changes to the LATEX kernel 49
Guard against \unskip in tabular cells 49
Fix Unicode table data 49
Improve \InputIfFileExists’s handling of file names 49
Improve interface for cross-references 49
Improve wording of a warning message 49
Avoid bad side-effects of \DeclareErrorFont. 49
nfssfont: Make font table generation the default action 49
trace: Add package support in the kernel 49

Changes to packages in the tools category 50
array: Warn if primitive column specifiers are overwritten 50
multicol: Introduce minrows counter for balancing 50
variosref: Better support for cleveref 50
xr: Support citations to bibliographies in external documents 50

Changes to packages in the amsmath category 50
amsmath: Introduce \overunderset command 50

Documentation updates 50
Highlighting the standard NFSS codes for series 50
LATEX base and doc distribution reunited 50

LATEX-dev formats now available

We know that many of you, especially developers and maintainers of important packages, have a strong interest in a stable LATEX environment.

In order to keep LATEX very stable for users whilst allowing for further development to continue, we now have a development branch of LATEX on GitHub containing development code for the upcoming release. When this code is ready for wider consumption and testing, we generate a pre-release of LATEX from this development branch and make it available on CTAN.

For users of the TeX Live and MiKTeX distributions it is therefore now straightforward to test their documents and code against the upcoming LATEX release with ease, simply by selecting a different program name (when using the command line) or by selecting a menu entry (after setting it up; see below).

If you do this then the latest version of the LATEX development format will be used to process your document, allowing you to test the upcoming release with your own documents and packages. For example, if you run

\texttt{pdflatex-dev \textit{myfile}}

then you will be greeted on the screen with something like \texttt{LaTeX2e <2019-10-01> pre-release-2} (identifying the pre-release format) instead of the normal \texttt{LaTeX2e <2018-12-01>}. In this pre-release you will find the latest new features that we have developed.

Our hopes

We don’t expect everybody to start using the development formats to participate in testing, but we hope that people with a strong interest in a stable LATEX environment (especially developers and maintainers of important packages) will use the new facilities and help us to ensure that future public releases of LATEX do not (as has happened in the past) require some immediate patches because of issues that were not identified by our internal regression test suite or by other testing we do.

Any issue identified when using the development format should preferably be logged as an issue on GitHub, following the procedure outlined on our website at \url{https://www.latex-project.org/bugs/} including the use of the latexbug package as described.

Our bug reporting process normally states that issues involving third-party software are out of scope as we can’t correct external packages; see [1]. However, in the particular case of the development format showing an incompatibility with a third-party package, it is fine to open an issue with us (in addition, please, to informing the maintainer of that package) so that we know about the problem and can jointly work on resolving it.

LATEX News, and the LATEX software, are brought to you by the LATEX Project Team; Copyright 2019, all rights reserved. –47
Details please . . .

More details and some background information about the concepts and the process are available in an upcoming TUGboat article: “The \LaTeX release workflow and the \LaTeX dev formats” [2].

Setting up menu items

While the command line call works out of the box if you have a recent \TeX Live or MiK\TeX installation, its use within an integrated editing environment doesn’t at this point in time (maybe the developers of these editors will include it in the future). However, it is normally fairly simple to enable it as most (or even all?) of them provide simple ways to call your own setup. How this works in detail depends very much on the environment you use, so we can’t give much help here.

But as an example: to provide an additional menu entry for XeLa\TeX-dev on a MacBook all that is necessary is to copy the file XeLa\TeX.engine to XeLa\TeX-dev.engine and change the call from xelatex to xelatex-dev inside.

Improving Unicode handling in pdf\TeX

Perhaps the most important improvement in this release is even better support for UTF-8 characters when using pdf\TeX.\(^1\)

When using a “Unicode engine”, any Unicode character (that is not acting as a command, i.e., is not “active”) can be used as part of the \texttt{\label/\ref} mechanism or can be displayed in a message or written to a file. In 8-bit engines, however, this was severely restricted: essentially you had to limit yourself to using ASCII letters, digits and a few punctuation symbols. With the new release, most of these restrictions have been removed and you now can write labels such as

\begin{verbatim}
\label{eq:größer}
\end{verbatim}

or use accented characters, etc., as part of a \texttt{\typeout} message. The only requirement remaining is that only those UTF-8 characters that are also available for typesetting can be used, i.e., only those characters for which adequate font support is loaded. Otherwise you will get an error message stating that the particular Unicode character is not set up for use with \LaTeX.X.

Note, however, that the restrictions on what characters can be used in the names of commands have not changed.

What is not possible when using an 8-bit engine such as pdf\TeX is to use characters other than ASCII letters as part of a command name. This is due to the fact that all other characters in such engines are not single character tokens, but in fact consist of a sequence of bytes and this is not supported in command names.

Improving file name handling in pdf\TeX

A related change is that file names used as part of \texttt{\input}, \texttt{\includegraphics}, etc., commands can now contain any Unicode characters allowed by the file system in use, including spaces. In this case, even characters that can’t be typeset (due to lack of font support) can be used.

Improving the filecontents environment

The filecontents environment now supports an optional argument in which you can specify that it is allowed to overwrite an already existing file; by default nothing is written if a file with the given name exists anywhere in the search tree. An alternative name for this option is force. Even then the environment will refuse to write to the jobname.tex to avoid clobbering its own input file. However, if you use a different extension on your input file you could still overwrite it (there is no way to test for that).

There is also an option nosearch, which specifies that only the current directory is examined for an existing file, not the whole \TeX inputs tree. This is useful if you want to write a local copy of a standard system file. Finally, noheader prevents writing a preamble to the file (this is the same as using the star form of the environment).

Another change is that this environment is now allowed anywhere in the document, which means it provides everything (and more) of what the now obsolete filecontents package provided.

Making more user commands robust

In the early days of \LaTeX many commands were fragile, i.e., they needed \texttt{\protect} in front of them when used in places such as section headings and other “moving arguments”, etc. in \LaTeX.X many of these commands were made robust, but still a fairly large number remained unnecessarily fragile.

In this release of \LaTeX we have now made a lot more commands robust. There is a very small collection of commands that must stay fragile because their expansion (maybe partially) at just the right time is critical. Yet others are unlikely to ever be needed in a “moving argument”.

Doing this for \texttt{\begin} and \texttt{\end} was rather tricky as the standard mechanism with \texttt{\DeclareRobustCommand} doesn’t work here, at least not for \texttt{\end} as that needs to expand during typesetting without generating a \texttt{\relax} (from the \texttt{\protect}). Such a token would start a new row in table environments, such as \texttt{\tabular}, etc. Furthermore, some packages try to look into the definition of \texttt{\end} by expanding it several times. Thus expansion with \texttt{\expandafter} had to produce exactly the same result as before. But in the end we overcame

\(^1\)The Japanese engines e-p\TeX and e-up\TeX can’t use these features yet as they don’t support the primitive \texttt{\ifinfo}.

Work is under way to resolve this in the engines.
that hurdle too, so now environments are automatically robust if used in places like headings or \texttt{typeout} and so forth.

What hasn't been tackled yet is the redefinitions in \texttt{amsmath}: this package redefines a number of basic math constructs that are now robust, so that they become fragile again once the package is loaded. This area will be addressed in a followup release. (\texttt{github issue 123})

\section*{Other changes to the \LaTeX{} kernel}

\subsection*{Guard against \texttt{\unskip} in tabular cells}

If a \texttt{tabular} or \texttt{array} cell started with a command that started with an \texttt{\unskip} then centering the column broke because the stretching glue on the left got removed. The fix for this was to add a minuscule, and hence unnoticeable, additional space after the stretching space: removing this extra space causes no problems.

This change was also applied in the \texttt{array} package. (\texttt{github issue 102})

\subsection*{Fix Unicode table data}

U+012F which is “i with ogonek” produced a “dotless i with ogonek” by mistake. This has been corrected. (\texttt{github issue 122})

The Unicode slots 27E8 and 27E9 have been mapped to \texttt{\textlangle} and \texttt{\textrangle} which is the recommended mapping. In the past they raised a \LaTeX{} error. (\texttt{github issue 110})

When doing cut-and-paste from other documents or websites, f-ligatures and others ligatures might end up as single Unicode characters in your file. In the past those got rejected by \LaTeX{}. We now define those Unicode slots and map them back to the sequence of individual characters constituting the ligature. If supported by the current font (which is normally the case) they are then reconstructed as ligatures and thus get typeset as desired. Otherwise they will come out as individual characters which is still better than an error message. (\texttt{github issue 154})

\subsection*{Improve \InputIfFileExists’s handling of file names}

In rare circumstances it was possible that \texttt{\InputIfFileExists} would work incorrectly, e.g., a construction such as

\begin{verbatim}
\InputIfFileExists{foo}{\input{bar}}{}
\end{verbatim}

would not load the files \texttt{foo.tex} and \texttt{bar.tex} but would load \texttt{bar.tex} twice. This has been corrected. (\texttt{github issue 109})

\subsection*{Improve interface for cross-references}

The packages \texttt{fncylab} and \texttt{varioref} provided a slightly improved definition of \texttt{\refstepcounter} which allowed the internal \texttt{\@\ldots} commands to receive the counter value as an argument, instead of acting as a simple prefix. This supports more complex formatting of the value in the reference.

These packages also provided the command \texttt{\labelformat} to help in the specification of such formatting in an easy way. For example, \texttt{\labelformat{equation}{eq.--(#1)}} specifies that references to equations automatically come out as “eq. (5)" or similar. As such a \texttt{\labelformat} declaration means a \texttt{\ref} command can no longer be successfully used at the start of a sentence, the packages also provided \texttt{\Ref} for such scenarios.

Both of these commands, \texttt{\labelformat} and \texttt{\Ref}, are now removed from the packages and instead made available in the kernel so there is no need to load additional packages.

\subsection*{Improve wording of a warning message}

The kernel now says “Trying to load …” instead of “Try loading …” in one of its informal messages to match style of similar messages. (\texttt{github issue 107})

\subsection*{Avoid bad side-effects of \DeclareErrorFont}

As a side effect of setting up the error font for NFSS, this declaration also changed the current font size back to 10pt. In most circumstances that doesn’t matter, because that declaration was meant to be used only during the format generation and not during a \LaTeX{} run. However, it has turned out to be used by some developers in other places (incorrectly in fact: e.g., inside some \texttt{.fd} files) where resetting the size causes havoc seemingly at random. The command has now changed to not produce such side effects. (\texttt{gnats issue latex/4399})

\subsection*{nfssfont: Make font table generation the default action}

With the small file \texttt{nfssfont.tex} it is possible to produce font tables and other font tests in the style set up by Don Knuth. In nearly all cases a font table is wanted, so this action has been made the default. Now one can simply hit enter instead of having to write \texttt{\table\bye}.

\subsection*{trace: Add package support in the kernel}

The \texttt{trace} package implements the commands \texttt{\traceon} and \texttt{\traceoff} that work like \texttt{\tracingall} but skip certain code blocks that produce a lot of tracing output. This is useful when debugging, to suppress uninteresting tracing from, for example, loading a font. Code blocks that should not be traced need to be surrounded by the commands \texttt{\conditionally@traceoff} and \texttt{\conditionally@traceon}.

The \LaTeX{} kernel now provides dummy definitions for these two commands so that package writers can use them in their packages regardless of \texttt{trace} being loaded or not.
Changes to packages in the tools category

array: Warn if primitive column specifiers are overwritten
With \texttt{newcolumntype} it is possible to define your own column specifiers for a \texttt{tabular} preamble; it is also possible to change existing ones. However, doing that for a primitive column specifier, such as \emph{c}, is seldom a good idea, since then its functionality becomes unavailable. The package was therefore supposed to warn the user in this case, but due to a missing \texttt{expandafter} in the code it never did—now it does. (github issue 148)

\texttt{multicol}: Introduce \texttt{minrows} counter for balancing
When there are only a few lines of text on a page at the end of a \texttt{multicols} environment, balancing the columns often looks rather odd: such as three columns each containing a single line. The balancing behavior can now be controlled through the counter \texttt{minrows} (default is 1) which specifies that, after balancing, there must be at least that many lines in the first column. Thus, if you set \texttt{minrows} to 2 then you would get a distribution of 2+1+0 lines and if set to three, the result would be 3+0+0 instead of the default 1+1+1.

What is most appropriate really depends on the circumstances, but this now gives you the tools to make local or global adjustments.

\texttt{varioref}: Better support for cleveref
The \texttt{varioref} package has been internally updated to provide better interfaces for packages such as \texttt{hyperref} and cleveref.

It also has a new package option \texttt{nospace} that stops \texttt{varioref} from meddling with space in front of its commands. The original behavior was always somewhat problematical and it is suggested that all new documents use this option (which should really have been the default).

Support was also added for the Arabic language through the option \texttt{arabic}.

\texttt{xr}: Support citations to bibliographies in external documents
The \texttt{xr} package can be used to cross-reference an external LATEX document. This means that even when a work is split over different documents (that need to be processed separately), \texttt{\ref} or \texttt{\pageref} can use labels from any document, creating links between them. This facility has now been extended so that \texttt{\cite} commands and their cousins can now also reference bibliographies in external documents; this feature was first provided in the package \texttt{xcite} by Enrico Gregorio.

Note that for technical reasons \texttt{xr} doesn’t work with \texttt{hyperref}. Use \texttt{xr-hyper} instead if you need the latter package.

Changes to packages in the amsmath category
\texttt{amsmath}: Introduce \texttt{\overset} and \texttt{\underset} command
The \texttt{amsmath} package has always offered the commands \texttt{\overset} and \texttt{\underset} to produce binary operators with something set above or below. But sometimes one needs to put something above and something below: The newly added \texttt{\overset} and \texttt{\underset} makes this easily possible.

Documentation updates
There are a number of documentation updates in files on the documentation page of the project website [4].

Highlighting the standard \texttt{nfss} codes for series
The Font Selection Guide [3] has been updated to strongly recommend that the standard codes should be used when providing font support. The reason for this recommendation is explained here.

The font selection scheme uses a number of standard codes for \texttt{\fontseries} and \texttt{\fontshape} to ensure that different fonts are comparable, e.g., that you get a “light” weight if you specify \texttt{1} and “extra bold” when you write \texttt{eb}, etc. Over the years people came up with a number of other creative short codes like \texttt{k}, \texttt{j}, \texttt{t} and others with the result that changing a font family required different codes and thus prevented users from easily mixing and matching different families. Some work has been undertaken to get back to a coherent scheme and all the font families supported through the program \texttt{autoinst} are now producing the standard codes again.

LATEX base and doc distribution reunited
For a long time the LATEX distribution available from CTAN was split into several parts to allow them to be uploaded or downloaded separately. As this is these days more confusing than helpful we have recombined the base part with the documentation part (as both are anyway always updated together). Thus the package \texttt{latex-doc} is no longer separately available from CTAN but contained in the \texttt{latex-base} distribution.

References
Experiences with the \LaTeX\ -dev formats
As reported in the previous \LaTeX\ News, we have made a pre-release version of the \LaTeX\ kernel available as \LaTeX\-dev. Overall, the approach of having an explicit testing release has been positive: it is now readily available in \TeX\ systems and is getting real use beyond the team.

Concerning this release \ldots (Lua\LaTeX engine)
The current release has been tested by a number of people, and we have had valuable feedback on a range of the new ideas. This has allowed us to fix issues in several of the new features, as described below.

We wish to thank all the dedicated users who have been trying out the development formats, and we encourage others to do so. Pre-testing in this way does mean that, for the vast majority of users, problems are solved before they even appear!

Improved load-times for expl3
The \LaTeX\3 programming layer, expl3, has over the past decade moved from being largely experimental to broadly stable. It is now used in a significant number of third-party packages, most notably \texttt{xpars}, for defining interfaces in cases where no expl3 code is “visible”. In addition, most \LaTeX\ documents compiled using Xe\TeX\ or Lua\TeX\ load fonts, which is written using expl3.

The expl3 layer contains a non-trivial number of macros, and when used with the Xe\TeX\ and Lua\TeX\ engines, it also loads a large body of Unicode data. This means that even on a fast computer, there is a relatively large load time when using expl3.

For this release, the team have made adjustments in the \LaTeX\ 2\epsilon\ kernel to pre-load a significant portion of expl3 when the format is built. This is transparent to the user, other than the significant decrease in document processing time: there will be no “pause” whilst loading
the Unicode data files. Loading \texttt{expl3} in documents and packages can continue to be done as usual; eventually, it will be possible to omit

\texttt{\textbackslash RequirePackage{expl3}}

to support older formats, this is still recommended at present.

**Improvements to \LaTeX{}'s font selection mechanism (NFSS)**

*Extending the shape management in NFSS*

Over time, more and more fonts have become available for use with \LaTeX{}. Many such font families offer additional shapes such as small caps italic (scit), small caps slanted (scsl) or swash (sw). By using \texttt{\fontshape} those shapes can be explicitly selected. For the swash shapes there is also \texttt{\textswash} and \texttt{\textscsw} available.

In the original font selection implementation a request to select a new shape always overrode the current shape. With the 2020 release of \LaTeX{} this has changed and \texttt{\fontshape} can now be used to combine small capitals with italic, slanted or swash letters, either by explicitly asking for \texttt{scit}, etc., or by asking for italics when typesetting already in small caps, and so forth.

Using \texttt{\upshape} will still change italics or slanted back to an upright shape but will not any longer alter the small caps setting. To change small capitals back to upper/lower case you can now use \texttt{\ulcshape} (or \texttt{\textulc}) which in turn will not change the font with respect to italics, slanted or swash. There is one exception: for compatibility reasons \texttt{\upshape} will change small capitals back to upright (n shape), if the current shape is sc. This is done so that something like \texttt{\acsshape...\upshape} continues to work as before, but we suggest that you don’t use that deprecated method in new documents.

Finally, if you want to reset the shape back to normal you can use \texttt{\normalshape} which is a shorthand for \texttt{\upshape\ulcshape}.

The way that shapes combine with each other is not hardwired; it is customizable and extensible if there is ever a need for this. The mappings are defined through \texttt{\DeclareFontShapeChangeRule} and the details for developers are documented in \texttt{source2e.pdf}.

The ideas for this interface extension have been pioneered in \texttt{fontspec} by Will Robertson for Unicode engines, and in \texttt{fontaxes} by Andreas Bihmann and Michael Ummels for pdf\LaTeX{}; they are by now used in many font support packages.

*Extending the font series management in NFSS*

Many of the newer font families also come provided with additional weights (thin, semi-bold, ultra-bold, etc.) or several running widths, such as condensed or extra-condensed. In some cases the number of different values for series (weight plus width) is really impressive: for example, Noto Sans offers 36 fonts, from ultra-light extra condensed to ultra-bold medium width.

Already in its original design, NFSS supported 9 weight levels, from ultra-light (u1) to ultra-bold (ub), and also 9 width levels, from ultra-condensed (uc) to ultra-expanded (ux): more than enough, even for a font family like Noto Sans. Unfortunately, some font support packages nevertheless invented their own names, so in recent years you have been able to find all kinds of non-standard series names (k, i, j and others), making it impossible to combine different fonts successfully using the standard NFSS mechanisms.

Over the course of the last year a small number of individuals, notably, Bob Tennent, Michael Sharpe and Marc Penninga, have worked hard to bring this unsatisfactory situation back under control; so today we are happy to report that the internal font support files for more than a hundred font families are all back to following the standard NFSS conventions.

Combining them is now again rather nice and easy, and from a technical perspective they can now be easily matched; but, of course, there is still the task of choosing combinations that visually work well together.

In the original font selection implementation, a request to select a new series always overrode the current one. This was reasonable because there were nearly no fonts available that offered anything other than a medium or a bold series. Now that this has changed and families such as Noto Sans are available, combining weight and width into a single attribute is no longer appropriate. With the 2020 release of \LaTeX{}, the management of series therefore changed to allow independent settings of the weight and the width attributes of the series.

For most users this change will be largely transparent as \LaTeX{} offers only \texttt{\textbf} or \texttt{\bfseries} to select a bolder face (and \texttt{\mdseries} to return to a medium series): there is no high-level command for selecting a condensed face, etc. However, using the NFSS low-level interface it is now possible to ask for, say, \texttt{\textseries{c}\selectfont} to get a condensed face (suitable for a marginal note) and that would still allow the use of \texttt{\textbf} inside the note, which would select a bold-condensed face (and not a rather odd-looking bold-extended face in the middle of condensed type).

The expectation is that this functionality will be used largely by class and package designers but, given that the low-level NFSS commands are usable on the document level and that they are not really difficult to apply, there are probably also a number of users who will enjoy using these new possibilities that bring \LaTeX{} back into the premier league for font usage.

The ways in which the different series values combine with each other is not hardwired but is again
customizable and extensible. The mappings are defined through \DeclareFontSeriesChangeRule and the details for developers are documented in source2e.pdf.

Font series defaults per document family
With additional weights and widths now being available in many font families, it is more likely that somebody will want to match, say, a medium weight serif family with a semi-light sans serif family, or that with one family one wants to use the bold-extended face when \textbf is used, while with another it should be bold (not extended) or semibold, etc.

In the past this kind of extension was provided by Bob Tennent’s mweights package, which has been used in many font support packages. With the 2020 release of \LaTeX\ this feature is now available out of the box. In addition we also offer a document-level interface to adjust the behavior of the high-level series commands \textit, \textmd, and of their declaration forms \bfseries and \mdseries, so that they can have different effects for the serif, sans serif and typewriter families used in a document.

For example, specifying

\begin{verbatim}
\DeclareFontSeriesDefault[rm]{bf}{sb}
\DeclareFontSeriesDefault[tt]{md}{lc}
\end{verbatim}

in the document preamble would result in \textbf producing semi-bold (sb) when typesetting in a roman typeface. The second line says that the typewriter default face (i.e., the medium series md) should be a light-condensed face. The optional argument here can be either rm, sf or tt to indicate one of the three main font families in a document; if omitted you will change the overall document default instead. In the first mandatory argument you specify either md or bf and the second mandatory argument then gives the desired series value in NFSS nomenclature.

Handling of nested emphasis
In previous releases of \LaTeX, nested \textit commands automatically alternated between italics and upright. This mechanism has now been generalised so that you can now specify for arbitrary nesting levels how emphasis should be handled.

The declaration \DeclareEmphSequence expects a comma separated list of font declarations corresponding to increasing levels of emphasis. For example,

\begin{verbatim}
\DeclareEmphSequence{\itshape,\upshape}
\end{verbatim}

uses italics for the first, small capitals for the second, and italic small capitals for the third level (provided you use a font that supports these shapes). If there are more nesting levels than provided, \LaTeX\ uses the declarations stored in \vmsans (by default \ulcshape\upshape) for the next level and then restarts the list.

The mechanism tries to be “smart” by verifying that the given declarations actually alter the current font. If not, it continues and tries the next level—the assumption being that there was already a manual font change in the document to the font that is now supposed to be used for emphasis. Of course, this only works if the declarations in the list’s entries actually change the font and not, for example, just the color. In such a scenario one has to add \textbf to the entry, which directs the mechanism to use the entry, even if the font attributes appear to be unchanged.

Providing font family substitutions
Given that pdf\LaTeX\ can only handle fonts with up to 256 glyphs, a single font encoding can only support a few languages. The \texttt{T1} encoding, for example, does support many Latin-based scripts, but if you want to write in Greek or Cyrillic then you will need to switch encodings to \texttt{LGR} or \texttt{T2A}. Given that not every font family offers glyphs in such encodings, you may end up with some default family (e.g., Computer Modern) that doesn’t blend in with the chosen document font. For such cases NFSS now offers \DeclareFontFamilySubstitution, for example:

\begin{verbatim}
\DeclareFontFamilySubstitution{LGR}
{Montserrat-LF}{IBMPlexSans-TLF}
\end{verbatim}

tells \LaTeX\ that if you are typesetting in the sans serif font \texttt{Montserrat-LF} and the Greek encoding LGR is asked for, then \LaTeX\ should use \texttt{IBMPlexSans-TLF} to fulfill the encoding request.

The code is based on ideas from the \texttt{substitufont} package by Günter Milde, but the implementation is different.

Providing all text companion symbols by default
The text companion encoding TS1 was originally not available by default, but only when the textcomp package was loaded. The main reason for this was limited availability of fonts with this encoding other than Computer Modern; another was the memory restrictions back in the nineties. These days neither limitation remains, so with the 2020 release all the symbols provided with the textcomp package are available out of the box.

Furthermore, an intelligent substitution mechanism has been implemented so that glyphs missing in some fonts are automatically substituted with default glyphs that are sans serif if you typeset in \texttt{tsf} and monospaced if you typeset using \texttt{tmm}. In the past they were always taken from Computer Modern Roman if substitution was necessary.

This is most noticeable with \texttt{oldstylenums} which are now taken from TS1 so that you no longer get 1234 but 1234 when typesetting in sans serif fonts and 1234 when using typewriter fonts.
If there ever is a need to use the original (inferior) definition, then that remains available as \textlegacyoldstylenums; and to fully revert to the old behavior there is also \textUseLegacyTextSymbols. The latter declaration reverts \textoldstylenums and also changes the footnote symbols, such as \textdagger, \textparagraph, etc., to pick up their glyphs from the math fonts instead of the current text font (this means they always keep the same shape and do not nicely blend in with the text font).

With the text companion symbols as part of the kernel, it is normally no longer necessary to load the textcomp package, but for backwards compatibility this package will remain available. There is, however, one use case where it remains useful: if you load the package with the option error or warn then substitutions will change their behavior and result in a \LaTeX error or a \LaTeX warning (on the terminal), respectively. Without the package the substitution information only appears in the \texttt{.log} file. If you use the option quiet, then even the information in the transcript is suppressed (which is not really recommended).

**New alias size function for use in .fd files**

Most of the newer fonts supported in \TeX have been set up with the autoinst tool by Marc Penninga. In the past, this program set up each font using the face name chosen by that font’s designer, e.g., “\textit{regular}”, “\textbf{bold}”, etc. These face names were then mapped by substitution to the standard NFSS series names, i.e., “\textit{m}” or “\textbf{b}”. As a result one got unnecessary substitution warnings such as “Font T1/abc/bold/n not found, using T1/abc/b/n instead.”

We now provide a new NFSS size function, alias, that can and will be used by autoinst in the future. It provides the same functionality as the subst function but is less vocal about its actions, so that only significant font substitutions show up as warnings.

**Suppress unnecessary font substitution warnings**

Many sans serif fonts do not have real italics but usually only oblique/slanted shapes, so the substitution of slanted for italics is natural and in fact many designers talk about italic sans serif faces even if in reality they are oblique.

With nearly all sans serif font families, the \LaTeX\ support files therefore silently substitute slanted if you ask for \texttt{\itshape} or \texttt{\textit{.}}. This is also true for Computer Modern in T1 encoding but in OT1 you got a warning on the terminal even though there is nothing you can do about it. This has now been changed to an information message only, written to the \texttt{.log} file. (\texttt{github issue 172})

**Other changes to the \LaTeX kernel**

**UTF-8 characters in package descriptions**

In 2018 we made UTF-8 the default input encoding for \LaTeX but we overlooked the case of non-ASCII characters in the short package descriptions used in declarations, e.g., in the optional argument to \texttt{\ProvidesPackage}. They worked (sometimes) before, but the switch to UTF-8 made them always generate an error. This has been corrected. (\texttt{github issue 52})

**Fix inconsistent hook setting when loading packages**

As part of loading a package, the command \texttt{\package.sty-h@@k} gets defined. However, attempting to load a package a second time resulted in this hook becoming undefined again. Now the hook remains defined so that extra loading attempts do not change the state of \LaTeX (relevant only to package developers). (\texttt{github issue 198})

**Avoid spurious warning if LY1 is made the default encoding**

Making LY1 the default encoding, as is done by some font support packages, gave a spurious warning even if \texttt{\textcompdefault} was changed first. This was corrected. (\texttt{github issue 199})

**Ensure that \textbackslash\texttt{\textbackslash} remains robust**

In the last release we made most document-level commands robust, but \texttt{\textbackslash\textbackslash} became fragile again whenever \texttt{\textbackslash\raggedright} or similar textsetting was used. This has been fixed. (\texttt{github issue 203})

**Make math delimiters robust in a different way**

Making math delimiters robust caused an issue in some situations. This has been corrected. This also involved a correction to \texttt{amsmath}. (\texttt{github issue 251})

**Allow more write streams with filecontents in Lua\TeX**

Most \TeX\ engines only support a maximum of sixteen concurrently open write streams, and when those have been used up, then \texttt{filecontents} or any other code trying to open one will fail. In Lua\TeX\ more write streams are available and those can also now be utilised. (\texttt{github issue 238})

**Allow spaces in \texttt{\textbackslash\texttt{\textbackslash}} option list**

Leaving spaces or newlines in the option list prevented the options from being correctly recognized. This has been corrected. (\texttt{github issue 256})

**New reverselist Lua callback type**

A new callback type, \texttt{reverselist}, was added: \texttt{post\_mlist\_to\_hlist\_filter} and \texttt{post\_linebreak\_filter} are now of this type.
Changes to packages in the graphics category

Make color & graphics user-level commands robust
Some of the user-level commands in color, graphics and graphicx, such as \textcolor or \includegraphics, were still fragile so didn’t work in moving arguments. All of these are now robust. (github issue 208)

Changes to packages in the tools category

Fixed column depth in boxed multicols
The multicols environment was setting \maxdepth when splitting boxes; but, due to the way the internal interfaces of \LaTeXX are designed, it should have used \maxdepth instead. As a result, balanced boxed multicols sometimes ended up having different heights even if they had exactly the same content.

Ensure that multicols does not lose text
The multicols environment needs a set of consecutively numbered boxes to collect column material. The way those got allocated could result in disaster if other packages allocated most boxes below box 255 (which \TeX always uses for the output page). In the original implementation that problem was avoided because one could only allocate box numbers below 255, but nowadays the \LaTeXX allocation routine allows allocating box numbers both below and above 255. So the assumption that when asking for, say, 20 boxes you always get a consecutive sequence of 20 box register numbers became no longer true: some of the column material could end up in box 255, where it would get overwritten. This has now been corrected by allocating all necessary boxes with numbers above 255 whenever there aren’t enough lower-numbered registers available. (github issue 190)

Allow spaces in \hhline arguments
The \hhline command, which allows the specification of rule segments in \texttt{tabular} environments, now allows (but ignores) spaces between its tokens: so \hhline{::=} is now allowed and is equivalent to \hhline{:=:}. This matches similar token arguments in \LaTeXX such as the \texttt{h t p l} argument on floats. A similar change has been made to the extended \hhline command in the \texttt{colortbl} package. (github issue 242)

\LaTeX requirements on engine primitives
Since the finalization of \eps-\TeX in 1999, a number of additional ‘utility’ primitives have been added to pdf\TeX. Several of these are broadly useful and have been required by expl3 for some time, most notably \texttt{pdfstrcmp}. Over time, a common set of these ‘post-\eps-\TeX’ primitives have been incorporated into X\TeX and (u)p\TeX; they were already available in Lua\TeX.

A number of these additional primitives are needed to support new or improved functionality in \LaTeXX. This is seen for example in the improved UTF-8 handling, which uses \texttt{ifincsname}. The following primitive functionality (which in Lua\TeX may be achieved using Lua code) will therefore be \texttt{required} by the \LaTeXX kernel and core packages from the start of 2021:

- \expanded
- \ifincsname
- \ifpdfprimitive
- \pdfcreationdate
- \pdfelapsedtime
- \pdffiledump
- \pdffilename
- \pdffilesize
- \pdflastxpos
- \pdflastypos
- \pdfmdfivesum
- \pdfnormaldeviate
- \pdfpageheight
- \pdfpagewidth
- \pdfprimitive
- \pdfrandomseed
- \pdffiledump
- \pfresettimer
- \pdfsavestream
- \pdffilemoddate
- \pdfspace
- \pdfshellescape
- \pdfstrcmp
- \pdfuniformdeviate

For ease of reference, these primitives will be referred to as the ‘pdf\TeX utilities’. With the exception of \expanded, these have been present in pdf\TeX since the release of version 1.40.0 in 2007; \expanded was added for \TeX Live 2019. Similarly, the full set of these utility primitives has been available in X\TeX from the 2019 \TeX Live release, and has always been available in Lua\TeX (some by Lua emulation). The Japanese p\TeX and up\TeX gained all of the above (except \texttt{ifincsname}) for \TeX Live 2019 and will both have that primitive also from the 2020 release onward.

At the same time, engines which are fully Unicode-capable must provide the following three primitives:

- \texttt{\Uchar}  
- \texttt{\Ucharcat}  
- \texttt{\Umathcode}

Note that it has become standard practice to check for Unicode-aware engines by using the existence of the \texttt{\Umathcode} primitive. As such, this is already a requirement: engines lacking these primitives cannot use Unicode features of the \LaTeXX 2e kernel or expl3. Note also that up\TeX can handle Unicode but it is not classed as a Unicode engine by the base \LaTeXX code.

References

https://latex-project.org/publications/

[2] \LaTeX Project Team: \texttt{\LaTeX} 2e font selection.
https://latex-project.org/help/documentation/fntguide.pdf

[3] \LaTeX documentation on the \texttt{\LaTeX} Project Website.
https://latex-project.org/help/documentation/
 QStringLiteral

In the past, many important new features are the kernel support for \texttt{xparse} and the introduction of the new hook management system for LaTeX, but as you can see there are many smaller enhancements and bug fixes added to the kernel and various packages.

### Providing \texttt{xparse} in the format

The \texttt{\texttt{xparse}} kernel for creating document-level commands has always been \texttt{newcommand}. This was a big step forward from LaTeX 2.09. However, it was still very limited in the \texttt{\texttt{xparse}} and the introduction of the new hook management system for LaTeX, but as you can see there are many smaller enhancements and bug fixes added to the kernel and various packages.

### Providing \texttt{xparse} in the format

The \texttt{\texttt{xparse}} kernel for creating document-level commands has always been \texttt{newcommand}. This was a big step forward from LaTeX 2.09. However, it was still very limited in the \texttt{\texttt{xparse}} and the introduction of the new hook management system for LaTeX, but as you can see there are many smaller enhancements and bug fixes added to the kernel and various packages.

### Providing \texttt{xparse} in the format

The official interface in the LaTeX kernel for creating document-level commands has always been \texttt{newcommand}. This was a big step forward from LaTeX 2.09. However, it was still very limited in the \texttt{\texttt{xparse}} and the introduction of the new hook management system for LaTeX, but as you can see there are many smaller enhancements and bug fixes added to the kernel and various packages.

### Providing \texttt{xparse} in the format

The official interface in the LaTeX kernel for creating document-level commands has always been \texttt{newcommand}. This was a big step forward from LaTeX 2.09. However, it was still very limited in the \texttt{\texttt{xparse}} and the introduction of the new hook management system for LaTeX, but as you can see there are many smaller enhancements and bug fixes added to the kernel and various packages.
large number of document and package commands. It does this by providing a rich and self-consistent syntax to describe a wide range of interfaces seen in \LaTeX\ packages.

The ideas developed in \texttt{xpars} are now sufficiently well tested that the majority can be transferred into the \LaTeX\ kernel. Thus the following commands have been added

- \texttt{\NewDocumentCommand, \RenewDocumentCommand, \DeclareDocumentCommand, \ProvideDocumentCommand}
- \texttt{\NewExpandableDocumentCommand, \RenewExpandableDocumentCommand, \ProvideExpandableDocumentCommand, \DeclareExpandableDocumentCommand}
- \texttt{\NewDocumentEnvironment, \RenewDocumentEnvironment, \ProvideDocumentEnvironment, \DeclareDocumentEnvironment}
- \texttt{\BooleanTrue \BooleanFalse}
- \texttt{\IfBooleanTF, \IfBooleanT, \IfBooleanF}
- \texttt{\IfNoValueTF, \IfNoValueT, \IfNoValueF}
- \texttt{\IfValueTF, \IfValueT, \IfValueF}
- \texttt{\SplitArgument, \SplitList, \TrimSpaces, \ProcessList, \ReverseBoolean}
- \texttt{\GetDocumentCommandArgSpec, \GetDocumentEnvironmentArgSpec}

Most, but not all, of the argument types defined by \texttt{xpars} are now supported at the kernel level. In particular, the types \texttt{g/G, 1 and u} are not provided by the kernel code; these are deprecated but still available by explicitly loading \texttt{xpars}. All other argument types are now available directly within the \LaTeX\ kernel.

\textit{A hook management system for \LaTeX}\n
With the fall 2020 release of \LaTeX\ we provide a general hook management system for the kernel and for packages. This will allow packages to safely add code to various kernel and package hooks and if necessary define rules to reorder the code in the hooks to resolve typical package loading order issues. This hook system is written in the L3 programming layer and thus forms the first larger application within the kernel that makes use of the \LaTeX\X3 functionality now available (if we discount \texttt{xpars} which has already been available for a long time as a separate package).

The file \texttt{lthook.dtx} holds the core management code for hooks and defines basic hooks for environments (as previously offered by \texttt{etoolbox}), \texttt{ltshipout.dtx} provides kernel hooks into the shipout process (making packages like \texttt{atbegshi}, etc., unnecessary) and the file \texttt{ltfilehook.dtx} holds redefinitions for commands like \texttt{\input} or \texttt{\usepackage} so that they offer hooks in a similar fashion to what is provided by the \texttt{filehook} package.

At the moment the integration is lightweight, overwriting definitions made earlier during format generation (though this will change after more thorough testing). For that reason the documentation isn’t in its final form either and you have to read through three different documents:

- \texttt{lthooks-doc.pdf} Core management interface and basic hooks for environments provided by the kernel.
- \texttt{ltshipout-doc.pdf} Hooks accessible while a page is being shipped out.
- \texttt{ltfilehook-doc.pdf} Hooks used when reading a file.

For those who wish to also study the code, replace \texttt{-doc} with \texttt{-code}, e.g., \texttt{lthooks-code.pdf}. All documents should be accessible via \texttt{texdoc}, e.g., \texttt{texdoc lthooks-doc} should open the core documentation for you.

\textit{Other changes to the \LaTeX\ kernel}\n
\texttt{symbol in math mode for large Unicode values}\n
The \LaTeX\X2E kernel defines the command \texttt{symbol}, which allows characters to be typeset by entering their \texttt{slot number}'. With the \LaTeX{} and \XeLaTeX{} engines, these slot numbers can extend to very large values to accommodate Unicode characters in the upper Unicode planes (e.g., bold mathematical capital A is slot number \texttt{1D400} in hex or \texttt{119808} in decimal). The \XeLaTeX{} engine did not allow \texttt{symbol} in math mode for values above \texttt{16}, this limitation has now been lifted. (\texttt{github issue 124})

\texttt{Correct Unicode value of \texttt{\textbar y} (\textbar y)}\n
The Unicode slot for \textbar y was incorrectly pointing to the slot for \textbar Y. This has been corrected. (\texttt{github issue 326})

\texttt{Add support for Unicode soft hyphens}\n
For a long time, the UTF-8 option for \texttt{inputenc} made the Unicode soft hyphen character (U+00AD) an alias for the \LaTeX\ soft hyphen \texttt{\-}. The Unicode engines \XeLaTeX{} and \LuaLaTeX{} behaved differently though: They either ignored U+00AD or interpreted it as an unconditional hyphen. This inconsistency is fixed now and \LaTeX\ always treats U+00AD as \texttt{\-}. (\texttt{github issue 323})

\texttt{Fix capital accents in Unicode engines}\n
In Unicode engines the capital accents such as \texttt{\capitalized}, etc., have been implemented as trivial shorthands for the normal accents (because other than Computer Modern virtually no fonts support them), but that failed when \texttt{hyperref} got loaded. This has been corrected. (\texttt{github issue 332})
**Support calc in various kernel commands**

The \hspace, \vspace, \addvspace, \ and other commands simply passed their argument to a \TeX primitive to produce the necessary space. As a result it was impossible to specify anything other than a simple dimension value in such arguments. This has been changed, so that now calc syntax is also supported with these commands.  

(github issue 152)

**Support \epsilon-\TeX length expressions in picture coordinates**

Picture mode coordinates specified with (\_,\_) previously accepted multiples of \unitlength. They now also allow \epsilon-\TeX length expressions (as used by the \texttt{glueexpr} primitive although all uses in picture mode are non-stretchy).

So, valid uses include \texttt{\put(2,2)} as previously, but now also uses such as \texttt{\put(0.5cm,0.4\textwidth)}.

Note that you can only use expressions with lengths; \texttt{\put(1+2,0)} is not supported.

**Spaces in filenames of included files**

File names containing spaces lead to unexpected results when used in the commands \texttt{\include} and \texttt{\includeonly}. This has now been fixed and the argument to \texttt{\include} can contain a file name containing spaces. Leading or trailing spaces will be stripped off, but spaces within the file name are kept. The argument to \texttt{\includeonly}, which is a comma-separated list of files to process, can also contain spaces with any leading and trailing spaces stripped from the individual filenames while spaces in the file names will remain intact.  

(github issues 217 and 218)

**Avoid extra line in \centering, \raggedleft or \raggedright**

If we aren’t justifying paragraphs then a very long word (longer than a line) could result in an unnecessary extra line in order to prevent a hyphen in the second-last line of the paragraph. This is now avoided by setting \texttt{\finalhyphenpenalties} to zero in unjustified settings.  

(github issue 247)

**Set a non-zero \baselineskip in text scripts**

As \texttt{\textsuperscript} and \texttt{\textsubscript} usually contain only a few characters on a single line the \texttt{\baselineskip} was set to zero. However, \texttt{\hyperref} uses that value to determine the height of a link box which consequently came out far too small. This has been adjusted.  

(github issue 249)

**Spacing issues when using \linethickness**

In some circumstances the use of \texttt{\linethickness} introduced a spurious space that shifted objects in a picture environment to the right. This has been corrected.  

(github issue 274)

**Better support for the legacy series default interface**

In the initial implementation of \LaTeX’s font selection scheme (NFSS) changes to any default were carried out by redefine some commands, e.g., \texttt{\seriesdefault}. In 2019 we introduced various extensions and with it new methods of customizing certain parts of NFSS, e.g., the recommended way for changing the series default(s) is now through \texttt{\DeclareFontSeriesDefault} \cite{1}. In this release we improved the support for legacy documents using the old method to cover additional edge cases.  

(github issues 306 and 315)

**Support for uncommon font series defaults**

If a font family was set up with fairly unusual font series defaults, e.g.,

\begin{verbatim}
\renewcommand{\ttdefault}{lmvtt}
\DeclareFontSeriesDefault{tt}{lm}
\end{verbatim}

then a switch between the main document families, e.g., \texttt{\ttfamily...\rmfamily} did not always correctly continue typesetting in medium or bold series if that involved adjusting the values used by \texttt{\mdseries} or \texttt{\bfseries}. This has now been corrected.  

(github issue 291)

**Checking the current font series context**

Sometimes it is necessary to define commands that act differently when used in bold context (e.g., inside \texttt{\textbf}). Now that it is possible in \LaTeX to specify different “\texttt{bf}” defaults based for each of the three meta families (\texttt{rm, sf} and \texttt{tt}) via \texttt{\DeclareFontSeriesDefault}, it is no longer easy to answer the question “am I typesetting in a bold context?”. To help with this problem a new command was provided:

\begin{verbatim}
\IfFontSeriesContextTF{(context)}{\{true code\}}{\{false code\}}
\end{verbatim}

The \texttt{(context)} can be either \texttt{bf} (bold) or \texttt{md} (medium) and depending on whether or not the current font is recognized as being selected through \texttt{\bfseries} or \texttt{\mdseries} the \texttt{(true code)} or \texttt{(false code)} is executed. As an example

\begin{verbatim}
usepackage[bm] % (bold math)
newcommand{\vbeta}{\IfFontSeriesContextTF{bf}{%\{ensuremath[bm]{\beta}\}}%\{ensuremath{\beta}\}}
\end{verbatim}

This way you can write \texttt{\vbeta-isotopes} and if used in a heading it comes out in a bolder version.  

(github issue 336)

**Avoid spurious package option warning**

When a package is loaded with a number of options, say \texttt{X, Y} and \texttt{Z}, and then later another loading attempt was made with a subset of the options or no options, it was possible to get an error message that option \texttt{X} is not
known to the package. This obviously incorrect error was due to a timing issue where the list of available options got lost prematurely. This has now been fixed.

(\texttt{github issue 22})

\textbf{Adjusting \texttt{fleqn}}

In \texttt{amsmath} the \texttt{\textbackslash mathindent} parameter used with the \texttt{fleqn} design is a rubber length parameter allowing for setting it to a value such as \texttt{\textbackslash em minus \textbackslash em}, i.e., so that the normal indentation can be reduced in case of very wide math displays. This is now also supported by the \LaTeX\ standard classes.

In addition a compressible space between formula and equation number in the \texttt{equation} environment got added when the \texttt{fleqn} option is used so that a very wide formula doesn’t bump into the equation number.

(\texttt{github issue 252})

\textbf{Provide \texttt{\clap}}

\LaTeX\ has inherited \texttt{\rlap} and \texttt{\llap} from plain \TeX\ (zero-sized boxes whose content sticks out to the left or right, respectively) but there isn’t a corresponding \texttt{\clap} command that centers the material. This missing command was added by several packages, e.g., \texttt{mathtools}, and has now been added to the kernel.

\textbf{Fix to legacy math alphabet interface}

When using the \LaTeX\ 2.09 legacy math alphabet interface, e.g., \texttt{\$\mathsf{-1}\$} instead of \texttt{\$\mathsf{\text{-1}}\$}, an extra math Ord atom was added to the formula in case the math alphabet was used for the first time. In some cases this math atom would change the spacing, e.g., change the unary minus sign into a binary minus in the above example. This has finally been fixed.

(\texttt{gnats issue latex/3357})

\textbf{Added tests for format, package and class dates}

To implement compatibility code or to ensure that certain features are available it is helpful and often necessary to check the date of the format or that of a package or class and execute different code based on the result. For that, \LaTeX\ previously had only internal commands (\texttt{\textbackslash ifpackage\texttt{\ldots}} and \texttt{\textbackslash ifclass\texttt{\ldots}}) for testing package or class names, but nothing reasonable for testing the format date. For the latter one had to resort to some obscure command \texttt{\textbackslash if\texttt{\ldots}} that, given its cryptic name, was clearly never intended for use even in package or class code. Furthermore, even the existing interface commands were defective as they are testing for “equal or later” and not for “later” as their names indicate.

We have therefore introduced three new CamelCase commands as the official interface for such tests

\begin{verbatim}
\IfFormatAtLeastTF{⟨date⟩}{⟨true code⟩}{⟨false code⟩}
\end{verbatim}

and for package and class tests

\begin{verbatim}
\IfPackageAtLeastTF{⟨package name⟩}{⟨date⟩}{⟨true code⟩}{⟨false code⟩}
\end{verbatim}

For compatibility reasons the legacy commands remain available, but we suggest to replace them over time and use the new interfaces in new code.

(\texttt{github issue 186})

\textbf{Avoid problematic spaces after \texttt{\textbackslash verb}}

If a user typed \texttt{\textbackslash verb\textbackslash l !- !\texttt{\textbackslash foo}} instead of \texttt{\textbackslash verb\textbackslash l ! \texttt{\textbackslash foo}} by mistake, then surprisingly the result was “\textbackslash l\texttt{\textbackslash foo}” without any warning or error. What happened was that the \texttt{\textbackslash l} became the argument delimiter due to the rather complex processing done by \texttt{\textbackslash verb} to render verbatim.

This has been fixed and spaces directly following the command \texttt{\textbackslash verb} or \texttt{\textbackslash verb*} are now ignored as elsewhere.

(\texttt{github issue 227})

\textbf{Provide a way to copy robust commands…}

With the previous \LaTeX\ 2.2 release, several user-level commands were made robust, so the need for a way to create copies of these commands (often to redefine them) increased, and the \LaTeX\ kernel didn’t have a way to do so. Previously this functionality was provided in part by Heiko Oberdiek’s \texttt{letltxmacro} package, which allows a robust command \texttt{\textbackslash foo} to be copied to \texttt{\textbackslash bar} with \texttt{\textbackslash LetLtxMacro\textbackslash bar\textbackslash foo}.

From this release onwards, the \LaTeX\ kernel provides \texttt{\textbackslash NewCommandCopy} (and \texttt{\textbackslash Renew\ldots} and \texttt{\textbackslash Declare\ldots} variants) which functions almost like \texttt{\textbackslash LetLtxMacro}. To the end user, both should work the same way, and one shouldn’t need to worry about the definition of the command: \texttt{\textbackslash NewCommandCopy} should do the hard work.

\texttt{\textbackslash NewCommandCopy} knows about the different types of definitions from the \LaTeX\ kernel, and also from other packages, such as \texttt{xparse}’s command declarations like \texttt{\textbackslash NewDocumentCommand}, and \texttt{etoolbox}’s \texttt{\textbackslash newrobustcmd}, and it can be extended to cover further packages.

(\texttt{github issue 239})

\textbf{... and a way to \texttt{\show} them}

It is sometimes necessary to look up the definition of a command, and often one not only doesn’t know where that command is defined, but doesn’t know if it gets redefined by some package, so often enough looking at the source doesn’t help. The typical way around this problem is to use \texttt{\textbackslash show}’s \texttt{\show primitive} to look at the definition of a command, which works fine until the command being \texttt{\show} is robust. With \texttt{\show\textbackslash frac} one sees

\begin{verbatim}
> \frac\texttt{\textbackslash macro}:
~->\texttt{protect \frac}
\end{verbatim}
which is not very helpful. To show the actual command the user needed to notice that the real definition of \textbackslash frac is in the \textbackslash frac\_c macro and do \textbackslash expandafter\textbackslash show\textbackslash cname frac\textbackslash space\textbackslash endcsname.

But with the machinery for copying robust commands in place it is already possible to examine a command and detect (as far as a macro expansion language allows) how it was defined. \textbackslash ShowCommand knows that and with \textbackslash ShowCommand\textbackslash frac the terminal will show

\begin{verbatim}
> \frac=robust macro:
    \protect \frac.
> \frac =long macro:
    #1#2->{\begingroup #1\endgroup \over #2}.
\end{verbatim}

(github issue 337)

Merge l3docstrip into docstrip
The file l3docstrip.tex offered a small extension over the original docstrip.tex file supporting the \textbackslash<@\textbackslash module> syntax of expl3. This has been merged into docstrip so that it can now be used for both traditional .dtx files and those containing code written in the L3 programming layer language. (github issue 337)

Support vertical typesetting with doc
The macrocode environment uses a \texttt{trivlist} internally and as part of this sets up the \texttt{@labels} box to contain some horizontal skips, but that box is never used. As a result this generates an issue in some circumstances if the typesetting direction is vertical. This has now been corrected to support such use cases as well. (github issue 344)

Record the counter name stepped by \texttt{\refstepcounter}
\texttt{\refstepcounter} now stores the name of the counter in \texttt{@currentcounter}. This allows packages like \texttt{zref} and \texttt{hyperref} to store the name without having to patch \texttt{\refstepcounter}. (github issue 300)

Native Lua\TeX{} behavior for \textbackslash-
\LaTeX{} changes \textbackslash- to add a discretionary hyphen even if \textbackslash hyphenchar is set to \textbackslash-1. This change is not necessary under Lua\TeX{} because there \textbackslash- is not affected by \textbackslash hyphenchar in the first place. Therefore this behavior has been changed to ensure that Lua\TeX{}’s (language specific) hyphenation characters are respected by \textbackslash-.

Allow \texttt{\par commands inside \textbackslash typeout}
\texttt{\typeout} used to choke when seeing an empty line or a \texttt{\par} command in its argument. However, sometimes it is used to display arbitrary user input or code (wrapped, for example, in \texttt{\unexpanded}) which may contain explicit \texttt{\par} commands. This is now allowed. (github issue 333)

Spacing commands moved from amsmath to the kernel
Originally \LaTeX{} only provided a small set of spacing commands for use in text and math; some of the commands like \textbackslash; were only supported in math mode. amsmath normalized and provided all of them in text and math. This code has now been moved to the kernel so that it is generally available.

<table>
<thead>
<tr>
<th>command name(s)</th>
<th>math</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textbackslash,</td>
<td>\textbackslash thinspace</td>
<td>x</td>
</tr>
<tr>
<td>\textbackslash!</td>
<td>\textbackslash negthinspace</td>
<td>xx</td>
</tr>
<tr>
<td>\textbackslash:</td>
<td>\textbackslash medspace</td>
<td>x</td>
</tr>
<tr>
<td>\textbackslashnegmedspace</td>
<td>xx</td>
<td>xx</td>
</tr>
<tr>
<td>\textbackslash;</td>
<td>\textbackslash thickspace</td>
<td>x</td>
</tr>
<tr>
<td>\textbackslash negthickspace</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>

(github issue 303)

Access raw glyphs in Lua\TeX{} without reloading fonts
\LaTeX{}‘s definitions for \textbackslash textquotesingle, \textbackslash textasciigrave, and \textbackslash textquotedbl for the TU encoding in Lua\TeX{} need special handling to stop the shaper from replacing these characters with curly quotes. This used to be done by reloading the current font without the \texttt{tlig} feature, but that came with multiple disadvantages: It behaves differently than the corresponding Xe\TeX{} code and it is not very efficient. This code has now been replaced with an implementation which injects a protected glyph node which is not affected by font shaping. (github issue 165)

Added a fourth empty argument to \texttt{\addcontentsline}
\LaTeX{}‘s \texttt{\addcontentsline} writes a \texttt{\contentsline} command with three arguments to the \texttt{.toc} and similar files. \texttt{hyperref} redefines \texttt{\addcontentsline} to write a fourth argument. The change unifies the number of arguments by writing an additional empty brace group. (github issue 370)

Lua\TeX{} callback \texttt{new\_graf} made exclusive
Corrected an incorrect callback type which caused return values from the \texttt{new\_graf} callback to be ignored and paragraph indentation to be suppressed. In the new version, only one \texttt{new\_graf} callback handler can be active at a time, which allows this handler to take full control of paragraph indentation. (github issue 188)

Changes to packages in the graphics category

Generate a warning if existing color definition is changed
If a color is defined twice using \texttt{\DefineNamedColor}, no info text \texttt{Redefining color ... in named color model ...} was written to the log file, because of a typo in the check. This has been corrected. (gnats issue graphics/3635)
Specifying viewport in the graphics package

Specifying a BoundingBox does not really have meaning when including non-EPS graphics in pdfTeX and LuaTeX. For some years the `graphicx` package `bb` key has been interpreted (with a warning) as a `viewport` key. This feature has been added to the two-argument form of \includegraphics, which is mostly used in the `graphics` package. \includegraphics*[viewport=1 2 3 4]{file} will now be interpreted in pdfTeX and LuaTeX in the same way as `graphicx`'s `\includegraphics[viewport=1 2 3 4]{file}`.

Normalizing `\endlinechar`

If `\endlinechar` is set to \texttt{-1} so that ends of lines are ignored in special contexts, then a low level TeX error would be generated by code parsing BoundingBox comments. The package now locally sets `\endlinechar` to its standard value while reading files. (\href{https://github.com/latex3/latex-base/commit/788d7f6}{\texttt{github issue 286}})

Files with multiple parts

Sometimes one has a graphics file, say, `file.svg`, and converts it to another format to include it in \LaTeX{} and ends up with a file named `file.svg.png`. In previous releases, if the user did `\includegraphics{file.svg}`, an error would be raised and the graphics inclusion would fail due to the unknown `svg` extension. The `graphics` package now checks if the given extension is known, and if it doesn't, it tries appending the known extensions until it finds a graphics file with a valid extension, otherwise it falls back to the file as requested. (\href{https://github.com/latex3/latex-base/issues/322}{\texttt{github issue 322}})

Changes to packages in the tools category

array: Support stretchable glue in \texttt{w}-columns

If stretchable glue, e.g., `\dotfill`, is used in \texttt{tabular} columns made with the `array` package, it stretches as it would in normal paragraph text. The one exception was \texttt{w}-columns (but not \texttt{w}-columns) where it got forced to its nominal width (which in case of `\hfill` or `\dotfill` is \texttt{0pt}). This has been corrected and now \texttt{w}-columns behave like all other column types in this respect. (\href{https://github.com/latex3/latex-base/issues/270}{\texttt{github issue 270}})

array: Use math mode for \texttt{w} and \texttt{w}-cells in \texttt{array}

The \texttt{w} and \texttt{w}-columns are LR-columns very similar to \texttt{1}, \texttt{c} and \texttt{r}. It is therefore natural to expect their cell content to be typeset in math mode instead of text mode if they are used in an \texttt{array} environment. This has now been adjusted. Note that this is a breaking change in version v2.5! If you have used \texttt{w} or \texttt{w}-columns in older documents either add `\$\{\}$\ldots<\$\}$` for such columns or remove the \$ signs in the cells. Alternatively, you can roll back to the old version by loading `array` with

```
\usepackage[array]{array}[v2.4]
```

in such documents. (\href{https://github.com/latex3/latex-base/commit/82f8a2d}{\texttt{github issue 297}})

array: Fix for `\firsthline` and `\lasthline`

Replacing `\hline` with `\firsthline` or `\lasthline` could lead in some cases to an increase of the tabular width. This has now been corrected. (\href{https://github.com/latex3/latex-base/issues/322}{\texttt{github issue 322}})

varioref: Support Japanese as a language option

The package now recognizes \texttt{japanese} as a language option. The extra complication is that for grammatical reasons `\vref`, `\Vref`, `\vrefrange` and `\fullref` need a structure different from all other languages currently supported. To accommodate this, `\vrefformat`, `\Vrefformat`, `\vrefrangeformat`, and `\fullrefformat` have been added to all languages. (\href{https://github.com/latex3/latex-base/issues/352}{\texttt{github issue 352}})

xr: Support for spaces in filenames

The command `\externaldocument`, provided by \texttt{xr}, now also supports filenames with spaces, just like `\include` and `\includeonly`. (\href{https://github.com/latex3/latex-base/issues/223}{\texttt{github issue 223}})

Changes to packages in the amsmath category

Placement corrections for two accent commands

The accent commands `\dddot` and `\ddddot` (producing triple and quadruple dot accents) moved the base character vertically in certain situations if it was a single glyph, e.g., `\$\dddot{Q}\$` were not at the same baseline. This has been corrected. (\href{https://github.com/latex3/latex-base/commit/4aa9c35}{\texttt{github issue 126}})

Fixes to \texttt{aligned} and \texttt{gathered}

The environments `\aligned` and `\gathered` have a trailing optional argument to specify the vertical position of the environment with respect to the rest of the line. Allowed values are `t`, `b` and `c` but the code only tested for `b` and `t` and assumed anything else must be `c`. As a result, a formula starting with a bracket group would get mangled without warning—the group being dropped and interpreted as a request for centering. After more than 25 years this has now been corrected. If such a group is found a warning is given and the data is processed as part of the formula. (\href{https://github.com/latex3/latex-base/commit/e6553f0}{\texttt{github issue 5}})

Detect Unicode engines when setting `\std@minus` and `\std@equal`

\texttt{amsmath} now detects the Unicode engines and uses their extended commands to define `\std@minus` and `\std@equal`. This avoids a package like \texttt{unicode-math} having to patch the code in the begin document hook to change the commands.

Use \LaTeX{} primitives where applicable

For a number of years \texttt{luatex-math} patched `\frac`, `\genfrac` and the `subarray` environment to make use of new \LaTeX{} primitives. This code has now been integrated into \texttt{amsmath}.
Changes to the babel package

Multilingual typesetting has evolved greatly in recent years, and babel, like \LaTeX{} itself, has followed the footsteps of Unicode and the W3C consortia to produce proper output in many languages.

Furthermore, the traditional model to define and select languages (which can be called “vertical”), based on closed files, while still the preferred one in monolingual documents, is being extended with a new model (which can be called “horizontal”) based on \emph{services} provided by babel, which allows defining and redefining locales with the help of simple ini files based on key/value pairs. The babel package provides about 250 of these files, which have been generated with the help of the Unicode Common Language Data Repository.

Thanks to the recent advances in lualatex and luaotfload, babel currently provides \emph{services} for bidi typesetting, line breaking for Southeast Asian and CJK scripts, nonstandard hyphenation (like ff to ff-f), alphabetic and additive counters, automatic selection of fonts and languages based on the script, etc. This means babel can be used to typeset a wide variety of languages, such as Russian, Arabic, Hindi, Thai, Japanese, Bangla, Amharic, Greek, and many others.

In addition, since these ini files are easily parsable, they can serve as a source for other packages.

For further details take a look at the babel package documentation [4].

References

[1] \LaTeX{} Project Team: \LaTeX{} 2ε news 31.  

[2] \LaTeX{} documentation on the \LaTeX{} Project Website.  
https://latex-project.org/help/documentation/

[3] \LaTeX{} issue tracker.  
https://github.com/latex3/latex2e/issues/

\emph{Babel—Localization and internationalization.}  
https://www.ctan.org/pkg/babel
Introduction

The focus of the June 2021 release is to provide further important building blocks for the future production of reliable tagged PDF output (see [1]); these enhancements are discussed in the next two sections.

Subsequent sections describe quite a number of recent smaller enhancements and fixes. As usual, more detail on individual changes can be found in the changes.txt files in the distribution and, of course, in the documented sources [2].

Extending the hook concept to paragraphs

Largely triggered by the need for better control of paragraph text processing, in particular when producing tagged PDF output, we have changed \LaTeX so that the kernel gains control both at the start and at the end of each paragraph. This is done in a manner that is (or should be) transparent to both packages and documents.

Besides the addition of internal control points for the exclusive use of the \LaTeX kernel, we also implemented four public hooks that can be used in packages or documents (via the normal hook management declarations) to achieve special effects, etc. Until now, such enhancements required redefinitions of \everypar or \par, which led to the usual issues since such changes can easily conflict with changes made by other packages.

The documentation of these new “paragraph hooks”, together with a few examples, is in \texdoc.pdf and,
for those who want to study it, the (quite interesting) code can be found in \texttt{ltpara-code.pdf}. Additionally, both of these files are included as part of the full kernel documentation in \texttt{source2e.pdf}.

\textbf{Extending the hook concept to commands}

Up to now, hook management covered hooks for only a few core areas, such as the hooks for the \texttt{\shipout} process or those in the \texttt{document} environment, as well as some “generic” hooks, both for file loading (helpful for patching such files) and for arbitrary environments (the hooks executed within \texttt{\begin} and \texttt{\end}). This concept of “generic hooks” has now been extended to provide \texttt{/before} and \texttt{/after} hooks for any (document-level) command—in theory at least.

In practice, these new generic \texttt{cmd} hooks, especially the \texttt{cmd/.../after}, hooks may fail with commands that are too complex to be automatically patched, breaking if the hook contains any code. These restrictions are documented in \texttt{ltpcmdhooks-doc.pdf}. However, given that these hooks are mainly meant for developers who wish to provide better interoperability between different packages, and between packages and the \LaTeX{} kernel, these restrictions are, we hope, of minor importance. Indeed, for commands where this mechanism can’t be applied, one is in the same situation as before; and for all others there will be a noticeable improvement.

These hooks will be especially important for our current project to provide accessible and tagged PDF output \cite{source2e.pdf} because we will eventually have to patch many third-party packages, and this must be done in controlled and standardized ways.

\textbf{Other hook business}

\textit{Shipping out a page while bypassing hooks}

In the 2020 October release, several hooks were added to control the process of constructing and shipping out a page box: these support, for example, the addition of background or foreground material to some or all pages.

We have now added a command, called \texttt{\RawShipout}, which does not do any rebuilding of the page box and so does not run most of these hooks. When using this new command, essential internal book-keeping is still carried out, such as updating the \texttt{totalpages} counter and adding \texttt{shipout/firstpage} or \texttt{shipout/lastpage} material when appropriate.

\textit{A new Lua callback in \texttt{ltshipout}, for custom attributes}

For use just before shipping out a page, there is now a \LaTeX{} callback \texttt{pre}\_\texttt{shipout}\_\texttt{filter} to contain final adjustments to the box being shipped out. This is particularly useful for \LaTeX{} packages which flag (using, for example, attributes or properties) elements on a page in order to apply effects (such as the insertion of “color commands”) to these elements at shipout.

\textit{Improved handling of file names}

\textbf{File names with spaces, multiple dots or UTF-8 characters}

In one of the recent \LaTeX{} releases we improved the interface for specifying file names so that they can now safely contain spaces (as is common these days), more than one dot character, and also UTF-8 characters outside the ASCII range. In the past this was only possible by applying a special syntax in the case of spaces, while file names with several dots often failed, as did most UTF-8 characters.

\textbf{Consequences for file names in \texttt{\include}}: \LaTeX{} has a built-in rule saying that you can normally leave out the extension if it is \texttt{.tex}. Thus \texttt{\input{file}} and \texttt{\input{file.tex}} both load \texttt{file.tex} (if it exists).

While this is convenient most of the time, it is a little awkward in some scenarios (for example, when both \texttt{file} and \texttt{file.tex} exist) and also when you manually try to implement the rule.

\LaTeX{} therefore had one special syntax for \texttt{\include} and \texttt{\includeonly}: they always expected that their arguments contain a file name\footnote{In the case of \texttt{\includeonly}, a comma-separated list of such names.} with no extension given, so that it had to be \texttt{.tex}. Thus, when you mistakenly wrote \texttt{\include{mychap.tex}} (for example, because you changed from \texttt{\input{file}} to \texttt{\include{file}}), \LaTeX{} went ahead and looked for the file \texttt{mychap.tex.tex} for inclusion and tried to use the file \texttt{mychap.tex.tex} for internal (auxiliary) information. The reason was that \texttt{\include} had to construct both of these file names from the given argument and it didn’t bother to do anything special with the supplied extension \texttt{.tex}.

With the new implementation this has changed: the extension \texttt{.tex} now gets removed/ignored if it was supplied. Thus \texttt{\include{mychap.tex}} now no longer looks for \texttt{mychap.tex.tex} but loads \texttt{mychap.tex} and uses \texttt{mychap.aux}.

\textbf{Normalization of robust commands in file names}

The handling of file names has been modified so that \texttt{\string} is applied to normalize robust commands within the file name. Previously, for example, \texttt{\input{\sqrt{2}}} would cause \LaTeX{} to loop indefinitely whereas with the new normalization it looks for the file named \texttt{sqrt 2.tex} (and therefore very likely reports “file not found”).

\textbf{Fix for filecontents with UTF-8 chars in the file name}

Since a few releases back, the \texttt{filecontents} environment has allowed UTF-8 characters in the file name. There was, however, a bug that would not allow \texttt{overwriting} a file with UTF-8 characters in its name. This has been fixed and now \texttt{filecontents} allows any characters in the file name.
Updates to the font selection scheme

A new hook in \selectfont

After \selectfont has changed the font, we now run a hook (selectfont) so that packages can make final adjustments. This functionality was originally provided by the everysec package but our implementation is slightly different and uses the standard hook management.

(github issue 444)

Change of font series/shape delayed until \selectfont

With the NFSS extensions introduced in 2020, the font series and shape settings can be influenced by changes to the font family. The settings of these two are now therefore delayed until \selectfont is executed; this avoids unnecessary or incorrect substitutions that may otherwise happen due to the order of declarations.

(github issue 444)

Glyphs, characters & encodings

Improved copy & paste for pdf\TeX\ documents

When compiling with pdf\TeX\ documents, additional information (from the file glyphhtounicode.tex) is now added automatically to the PDF file in order to improve copying from, and searching in, text.

In particular, this allows the most common ligatures to be copied as intended from all generated PDF files without the need to explicitly load the package cmap.

(github issue 444)

Support for more Unicode characters

\textcomp is quite capable of typesetting characters such as “\textasciimark”, but until now it could not access some Unicode characters from the Latin Extended Additional block. This meant that, for example, there were no Unicode mappings for some characters that are used to write Sanskrit words in Latin transliteration (as seen in books about yoga, Buddhist philosophy, etc.). These characters have now been added so that they can be entered directly instead of using \textasciimark, etc.

(github issue 448)

More “dashes” in encodings OT1, T1 and TU

When pasting in text from external sources, one can encounter these three Unicode characters "2011 (non-breaking hyphen), "2012 (figure dash) and "2015 (horizontal bar), in addition to the more common "2013 (en-dash) and "2014 (em-dash). In the past, these first three produced an error message when used with pdf\TeX\ (since they are not available in OT1 or T1 encoded fonts). They now typeset an approximation to the glyph: e.g., the “figure dash” is approximated by an en-dash.

With Unicode engines they either work (when the glyph is contained in the selected Unicode font) or they typeset nothing, producing a “Missing character” warning in the log file.

With all engines these characters can also now be accessed using the command names \textnonbreakinghyphen, \textfiguredash and \texthorizontalbar, respectively.

(github issue 404)

Poor man’s \textasteriskcentered

The \textasteriskcentered symbol, used as part of the set of footnote symbols in \textcomp, is assumed to be implemented by every font with the TS1 encoding (when pdf\TeX\ is used) or with the TU encoding for the Unicode engines. That assumption is unfortunately not correct for all fonts since, for example, the stix2 fonts don’t provide this glyph. A result is that one gets missing glyph messages when using \textasteriskcentered, etc.

Therefore \textasteriskcentered now checks whether there is such a glyph and, if not, uses a normal “*”, but slightly enlarged and lowered. This may not be perfect in all cases, but it is certainly better than no glyph showing up.

(github issue 502)

The characters from textcomp are in the kernel

A couple of releases back, the functionality of the textcomp package was integrated into the \textcomp kernel. Thus it is no longer necessary to load this package in order to access glyphs such as \textcopyright, \texteuro\ or \textyen.

At this time the opportunity was also taken to bring some order to the chaos surrounding the question: “which glyphs from the TS1 encoding are available in a given font?”. This was done using an approach based on font families and collections, with the differing glyph coverage of the ‘text symbols’ being indicated by assigning to a font family or collection a “sub-encoding number” that indicates which glyphs from the TS1 encoding are guaranteed to be available when using a font from that family or collection. This assignment ensures that \textcomp always errs on the side of caution, possibly claiming that a glyph is not available even when it in fact is.

A note on the history of “text symbols” and the TS1 encoding:

The “text symbol encoding” (TS1) was originally designed at the Cork Conference as a companion to the T1 encoding. In it various symbols that are not subject to hyphenation got assembled and the textcomp package was developed to make them accessible. Unfortunately the \textcomp community was a bit too enthusiastic and included several symbols only available in a few \textcomp fonts and some, such as the capital accents, not available at all but developed as part of the reference font implementation.

In hindsight that was a very bad idea because it meant that other existing fonts (at the time) and later new fonts that got developed were unable to provide the full set of glyphs that made up the TS1 encoding. For existing free PostScript fonts people took the extra effort
and produced virtual fonts that faked (some) of the missing glyphs. But this was and is a time-consuming effort so it was done for only a few basic fonts. But even then, only some fonts included all glyphs from textcomp into the L\TeX kernel. Unfortunately the chaos around this topic had increased with it.

When we recently integrated the functionality of the textcomp font into the \LaTeX kernel many new free fonts had appeared and unfortunately the chaos around the question "which glyphs of the TS1 encoding are implemented by which font" had increased with it. Not only did one find many new holes, it was next to impossible to order the set of fonts into a reasonable set of sub-encodings that are contained in each other in a single sequence.

In the end we decided on nine or ten sub-encodings with a reasonable number of fonts in each so that all fonts implemented all glyphs of the sub-encoding they got mapped to. Thus when typesetting with a font one could be sure that a command like \textcoyleft would either typeset the requested character (if the glyph was part of the sub-encoding the font belonged to) or it would raise an error, saying that the glyph is unavailable in that font. The mapping would ensure that \LaTeX always errs on the side of caution, because it might claim a glyph is unavailable even though in fact it is.

For example, the old pcr (PostScript Courier) font (as well as most other older PS fonts) is mapped to sub-encoding 5 and therefore claims that \textasciigrave is unavailable even though in fact for Courier this is not true. If one uses such a font and this becomes an issue then there are a couple (suboptimal) possibilities. For one, one can alter the mapping of Courier and pretend that belongs to a fuller sub-encoding, e.g.

\begin{verbatim}
\DeclareEncodingSubset{TS1}{pcr}{2}
\end{verbatim}

The downside is, that \LaTeX then believes other glyphs that are in fact unavailable are also there, so that it is important to check that the final document doesn’t have some missing glyphs.

An alternative is to pretend that \textasciigrave can always be taken from the TS1 encoding (no questions asked):

\begin{verbatim}
\DeclareTextSymbolDefault{\textasciigrave}{TS1}
\end{verbatim}

Again there is a danger that this is not true when it is used with a different font and would then generate a missing glyph.

Finally, and possibly the best solution, if not impossible for other reasons, is to simply use a different font, for example, to use the \TeX Gyre Cursor font (a reimplementation of Courier with a much more complete glyph set).

---

New or improved commands

Adjusting itemize labels with \labelitemfont

The command \labelitemfont was introduced already with the \LaTeX release 2020-02-02, but back then we forgot to describe it, so we do this now. Its purpose is to resolve some bad formatting issues with the itemize environment and also to make it easier to adjust the layout when necessary. What could happen in the past was that the itemize labels (e.g., the *) would sometimes react to surrounding font changes and could then suddenly change shape, for example to •.

This new command \labelitemfont, which defaults to \normalfont, can be used to provide additional control in the typesetting of each label. Thus by choosing different settings other effects can be achieved. Here are two examples:

\begin{verbatim}
\renewcommand\labelitemfont
{\normalfont\fontfamily{lmss}\selectfont}
\renewcommand\labelitemfont
{\normalfont\fontfamily{lmss}\selectfont}
\end{verbatim}

The first definition will take the symbols from the font Latin Modern Sans, so that you get •, –, • and ·; while the second variant freezes the font family and shape, but leaves the series as a variable quantity, so that an itemize in a bold context would show bolder symbols. Making \labelitemfont empty would give you back the buggy old behavior. (github issue 497)

Producing several marks for one footnote

It is sometimes necessary to reference the same footnote several times: i.e., to produce several footnote marks using the same number or symbol. This is now easily possible by placing a \label within the referenced \footnote and referencing this label by using the new command \footref. This means that footnote marks can be generated to refer to arbitrary footnotes (including those in minipages).

This \footref command has previously been available, but only when using certain classes or the footmisc package. (github issue 482)

Allow \nocite in the preamble

A natural place for \nocite would be the preamble of the document, but for historical reasons \LaTeX issued an error message if it was placed there. This command is now allowed in the preamble. (github issue 424)

Made \ generally robust

In 2018 most \LaTeX user-level commands were made robust, including the \ command. However, \ gets redefined in various environments and not all these cases were caught: such as, in particular, its use as the row delimiter in \texttt{tabular} structures. This has been corrected so that \ should now be robust in all circumstances.
This change also fixed one anomaly present in the past: in a \tabular preamble of the form
\{1x\{raggedright\}p{10cm}\r
a \& b \& c \& d \}
the column 2 would have the definition used within \raggedright and so it would not indicate the (premature) end of the \tabular. Thus, for example,
\begin{verbatim}
a & b1 & b2 & c \\
\end{verbatim}
was interpreted as a single row of the \tabular (as intended), whereas
\begin{verbatim}
a & b & c \\
\end{verbatim}
resulted in two rows! This happened because the \& directly following the \& got interpreted while it still had the “end the row” meaning and not yet the “start a new line within the second column” meaning.

With \& now being robust, the special scanning mode initiated by the & ends immediately when this command is seen: the second column is therefore then started, which results in the \& being interpreted as being within that column and hence as having its expected, within-column, meaning.

We have restored consistency here: now both of the above lines produce a single \tabular row. As before, you can put \raggedright\arraybackslash in the \tabular’s preamble for a column to ensure that \& is always interpreted as a tabular row separator when used in that column. And you can use \tabularnewline to explicitly ask for a new table row, even when \& has a different meaning within the current column.

(\texttt{github} issue 548)

\section{Allow extra space between name and address in letter class}

The \opening command in the letter class expects the name and address to be separated by \&, but it didn’t allow the use of an optional argument to add some extra space after the name. The code has now been slightly altered to allow this.

(\texttt{github} issue 427)

\section{Additions to \texttt{tracingall}}

In July 2020 David Jones suggested an extension to \TeX engines, that added the possibility to set \texttt{\let\tracinglostchars=3} in order to generate an error message in case some character is missing from a font. In previous years, a warning about a missing character was silently printed to the \texttt{.log} file (if \texttt{\tracinglostchars > 0}) and to the terminal (if \texttt{\tracinglostchars > 1}). This extension was added for \TeX Live and MiK\TeX (except in Knuth’s \TeX, of course), so that with \texttt{\tracinglostchars > 2} you now also get an error message for each missing glyph.

Later, in January 2021, Petr Olšák suggested yet another extension: a new primitive parameter \texttt{\tracingstaklevels} that, when both it and \texttt{\tracingmacros} are positive, will add to the tracing information for each macro a visual indication (using dots) of its nesting level in the macro expansion stack.

These changes have both now been added to \LaTeX’s debugging macros \texttt{\tracingall} and \texttt{\tracingnone}, so that these two new extensions are activated/deactivated as appropriate, so long as the \TeX engine supports them. An example document demonstrating these parameters is in the linked GitHub issue.

(\texttt{github} issue 524)

\section{Code improvements}

\begin{enumerate}
  \item \texttt{\par} at the end of \texttt{\marginpar}

Previously, \LaTeX ended a \texttt{\marginpar} without ever explicitly calling \texttt{\par}. This command is now explicitly added because it is essential to the correct working of the paragraph hooks.

Another case where this issue caused problems was the \texttt{\linenop} package, where the last line was not numbered if the \texttt{\marginpar} ended without an explicit \texttt{\par}.

(\texttt{github} issue 489)

  \item \texttt{\AtEndDocument} hook in vertical mode

Until now \texttt{\end{document}} executed the code from the \texttt{\AtEndDocument} hook as its first action. This meant that this hook was executed in horizontal mode if the user left no empty line after the last paragraph. As a result, one could get a spurious space added when, for example, that code contained a \texttt{\write} statement. This was fixed and now \texttt{\end{document}} first issues a \texttt{\par} to ensure that it always goes into vertical mode.

(\texttt{github} issue 385)

\end{enumerate}

\section{Color groups made permanent}

The use of color in certain \LaTeX constructs, especially boxes, needs an extra layer of grouping to ensure that the color setting does not \texttt{escape} and continue outside the box when it shouldn’t. To support this, the \LaTeX kernel defines a number of commands, e.g., \texttt{\color@begingroup} to be used in such places.

Until now, these commands were initially set as no-ops and only the color packages redefined them to become real groups; this methodology complicates the coding as one has to account for a group being present or not (depending on what is loaded in the document). The kernel therefore now permanently adds these “color groups”.

(\texttt{github} issue 488)

\section{Provide the raw option list to key/value option handlers}

Before any further processing of the option list, the original (un-normalized, “raw” and unchanged) list of package or class options is now saved, as \texttt{\@raw@opt@…}; this list is not used by the standard option processing code but it is now available for use by extended class/package processing systems. Note that, for compatibility reasons, the standard option processing code has not been changed.

One aspect of this change does affect the standard processing: any tokens to the right of an * sign are removed from consideration when constructing the
Changes to packages in the tools category

layout: Added language options
This package now recognizes \texttt{japanese} and \texttt{romanian} as language options.  

array and longtable: Make \texttt{\backslash generally robust}
The fix for this issue was also applied to these packages; see above.  

longtable: General bug fix update
This is a minor update to the longtable package that
fixes several reported bugs: notably the possibility of
incorrect page breaks when floats appear on the page
where a longtable starts. As this may affect page
breaking in existing documents, a rollback to longtable
4.13 (longtable-2020-01-07.sty) is supported.

\texttt{\traceon}
The \texttt{\tracingstacklevels} and \texttt{\tracinglostchars}
extensions to \texttt{\traceall} (see above) were also added
to \texttt{\traceon} in the \texttt{trace} package, so its users can
also benefit from these new debugging possibilities.  

bm: Better support for commands with optional arguments
Some uses of optional arguments in \texttt{\bm} stopped being
supported (in 2004) when \texttt{\textbf} was
used internally by the format instead of \texttt{\textbf}
This update handles both versions of this command and
restores the original behavior.

In addition, package options for guiding the use of
“poor man’s bold” in fallback situations were added.  

Changes to packages in the amsmath category
The fix for issue 548 was also applied in amsmath; see above.  

References
[1] Frank Mittelbach and Chris Rowley: \texttt{\LaTeX} Tagged
\texttt{PDF}—A blueprint for a large project.
https://latex-project.org/publications/
indexbyyear/2020/
[2] \texttt{\LaTeX} documentation on the \texttt{\LaTeX} Project Website.
https://latex-project.org/help/documentation/
[3] \texttt{\LaTeX} Project Team: \texttt{\LaTeX} 2ε \texttt{news 32}.
https://latex-project.org/news/latex2e-news/
ltnews32.pdf
Contents

Introduction 69

Hook business 69
  Provide \ActivateGenericHook 
  Standardized names for the generic hooks 
  Some file hooks made one-time 
  Clearing extra hook code for the next invocation 
  Cleaning up after \UseOneTimeHook 
  \RemoveFromHook with a missing code label 
  Patching commands with parameter tokens 
New or improved commands 71
  \NewCommandCopy and \ShowCommand extended 
  Undo math alphabet allocations if necessary 
  New default value for \tracinglostchars 
  \PackageNote and \ClassNote added 
  New \ShowFloat command 
  New argument for \counterwithin/without 
  Tests for package and class loading 
  Better handling for a misuse of \include 
Code improvements 72
  Use OpenType version of Latin Modern 
    Upright Italic font 
  Additional Extended Latin characters predefined 
  Check \endfoo in \NewDocumentEnvironment 
  Improve the error message \begin ended by 
  Pick up all arguments to \contentsline 
  Allow dropping a math list in Lua\TeX{} callback 
  Extended label handling in package code 
  Better message if text accent used in math mode 
Bug fixes 73
  Replicate argument processors for all 
  embellishments in command declarations 
  Correct case changing of \ij and \IJ 
  Legacy font series default changes 
  Use of # in \textbf and similar commands 
Changes to packages in the \texttt{amsmath} category 73
  Improved compatibility with hyperref 
Changes to packages in the \texttt{graphics} category 74
  graphicx: New key, for alt text 

Changes to packages in the \texttt{tools} category 74
  array: No \mathsurround around a tabular 
  longtable: Improvements after a section heading 
  multicol: Better column break control 
  varioref: Improved handling of missing labels 

Introduction

This release of \LaTeX{} does not contain any major new modules, but is focused around consolidation and improvements of the functionality introduced in previous releases. In addition, various smaller enhancements and bug fixes have been added to the kernel and the core packages.

Hook business

Since the introduction of the hook management system in the 2020 release of \LaTeX{} package developers have started to make more and more use of this new functionality. One result of this increased activity has been a number of queries which show that some of the documentation was not precise enough and that some clarifications were needed; these deficiencies have now been addressed in the documentation. The increased usage has also revealed a small number of errors that we thought should be corrected now, while the adoption rate is still relatively small; the following problems have therefore been addressed in this release.

Provide \ActivateGenericHook

The hook management system offers a number of generic hooks, i.e., hooks whose names contain a variable component such as the name of an environment. Predeclaring such hooks is not feasible, so these hooks use a different mechanism: they are implicitly available, springing into life the moment a package, or the document preamble, adds any code to one by using \AddToHook. The kernel offers such hooks for environments (\texttt{env/...}) and commands (\texttt{cmd/...}), and also for files, packages and classes (\texttt{file/...}, \texttt{include/...}, \texttt{package/...}, \texttt{class/...}).

It is also possible to offer such generic hooks in packages if, for example, hooks are needed that depend on the current language and therefore need the language name as part of the hook name (but you probably don’t know beforehand all the necessary names).

If you want to offer such generic hooks, you can now do this by using \UseHook or \UseOneTimeHook.
in your (package) code, but without declaring the hook with \NewHook. However, without further work, a call to \UseHook with an undeclared hook name will do nothing: so, as an additional setup step, it is necessary to explicitly activate the generic hook by using \ActivateGenericHook.\footnote{Note that in the previous release we offered \ProvideHook as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer \ActivateGenericHook instead.}

Assuming that you don’t know all the different hook names up front, it will remain the task of the users of your package to activate the hook themselves before adding code to it. For example, Babel offers hooks such as babel/language/afterextras that enable a user to add language specific declarations to these “extras”. One can then write

\begin{verbatim}
\ActivateGenericHook
{babel/ngerman/afterextras}
\AddToHook{babel/ngerman/afterextras}
{\color{blue}}
\end{verbatim}

after which all German words would be colored blue in the text.

Note that a generic hook produced in this way is always a normal hook.

**Standardized names for the generic hooks**

The initial set of generic hooks provided by the kernel had two patterns of names: ones like env/(name)/after, with the variable, ⟨name⟩, part in the middle position; and ones like file/after/(name), with the variable part in the third position. The coexistence of these two types caused confusion because the user had to remember in which position the variable part was supposed to go; and it also made the code more complicated and slower.

The file-related hooks have therefore been renamed so that the variable part of the name is in the middle, as with all other hooks. The changes are listed here:

<table>
<thead>
<tr>
<th>Old name</th>
<th>New name</th>
</tr>
</thead>
<tbody>
<tr>
<td>file/before/(name)</td>
<td>file/(name)/before</td>
</tr>
<tr>
<td>file/after/(name)</td>
<td>file/(name)/after</td>
</tr>
<tr>
<td>package/before/(name)</td>
<td>package/(name)/before</td>
</tr>
<tr>
<td>package/after/(name)</td>
<td>package/(name)/after</td>
</tr>
<tr>
<td>class/before/(name)</td>
<td>class/(name)/before</td>
</tr>
<tr>
<td>class/after/(name)</td>
<td>class/(name)/after</td>
</tr>
<tr>
<td>include/before/(name)</td>
<td>include/(name)/before</td>
</tr>
<tr>
<td>include/after/(name)</td>
<td>include/(name)/after</td>
</tr>
<tr>
<td>include/after/(name)</td>
<td>include/(name)/after</td>
</tr>
</tbody>
</table>

Since this is a breaking change, the old names will still work for a while so that users and package authors have enough time to adjust; but a warning will be issued when the old names are used. Eventually the deprecated names will be turned into errors and then removed completely. \footnote{Note that in the previous release we offered \ProvideHook as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer \ActivateGenericHook instead.}

---

**Some file hooks made one-time**

Classes, packages and included files can only be loaded once in a \LaTeX{} document. For this reason, the hooks that are specific to loading such files have been made one-time hooks. Beside being more efficient, this supports the following important use case:

\begin{verbatim}
\AddToHook{package/varioref/after}
{\cdots apply when the package gets loaded, or apply now (if it is already loaded) \cdots}
\end{verbatim}

without the need to first test whether the package is already loaded. \footnote{Note that in the previous release we offered \ProvideHook as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer \ActivateGenericHook instead.}

---

**Clearing extra hook code for the next invocation**

There are a few use cases where it is helpful if one can cancel an earlier use of \AddToHookNext: for example, when a page is discarded with \DiscardShipoutBox because only some pages of the document are printed. For such situations the new command \ClearHookNext is now provided. \footnote{Note that in the previous release we offered \ProvideHook as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer \ActivateGenericHook instead.}

---

**Cleaning up after \UseOneTimeHook**

Some hooks are meant to be used only once in a document, and any further attempt to add code to one of these will cause the code to be executed immediately instead of being added to the hook. The initial implementation of this concept was very simple and didn’t anticipate that packages may try to execute a one-time hook several times, resulting in the hook code being executed repeatedly. Thus the implementation was fine for simple cases (such as the \begindocument hook) but it causes trouble if the one-time hook was intended, for example, as an initialization hook that is used just once (when a command is first called) but is then ignored in further calls.

This deficiency has been addressed, and now a one-time hook will only be executed once, with its code being removed after use to free up some memory. \footnote{Note that in the previous release we offered \ProvideHook as a means to achieve this effect, but the name was badly chosen so we decided to deprecate it and now offer \ActivateGenericHook instead.}

---

**\RemoveFromHook with a missing code label**

In the first version of \RemoveFromHook, when the code label to be removed didn’t exist in the hook a “removal order” would be queued; and then, the next time something tried to add that label to the hook, this \AddToHook action would be cancelled by the removal order, so that no code would be added that one time. This was so that, in principle, package loading order wouldn’t matter. However, this implementation didn’t work as intended because, while two \AddToHook actions with a given label would be removed by a single \RemoveFromHook, one \RemoveFromHook could not cancel two \AddToHook actions for that label; this caused confusion and also led to further problems.

The implementation has now been changed, so that \RemoveFromHook removes only code labels that already
exist in a hook: it will display a warning if there is no such code label.

Note that, whereas when working with a single package you should use `\RemoveFromHook` to remove a code label, when working with more than one package, the `voids` relation should preferentially be used. This is best because this relation is non-destructive (meaning that it can be reverted later by using another relation), and it is also truly independent of package loading order. (github issue 625)

Patching commands with parameter tokens
In the last release, \endtext{}X's hook mechanism was extended to add support for hooking into commands using generic `cmd` hooks (see [5]). That version of the extension had a bug: the patching of some commands that contained a parameter token (normally `#`) in their definition would fail with a low-level `\endtext{}X` error. This has now been fixed so that patching now works for those commands as well. (github issue 697)

New or improved commands

\NewDocumentCommandCopy and \ShowCommand extended
Since the 2020-10-01 release (see [4]), \endtext{}X has provided \NewDocumentCommandCopy to copy robust commands, and \ShowCommand to show their definitions on the terminal. In that same release, the `xparse` package was integrated into the kernel (as \ltcmd) to offer \NewDocumentCommand, etc. However, the extended support for \NewDocumentCommandCopy and \ShowCommand was not implemented in \ltcmd. The present \endtext{}X release implements this support, so now commands defined with \NewDocumentCommand and friends can also be copied, and their definitions can be easily shown on the terminal without the need for \texttt{\csname gymnastics\endtext{}eex@\textit{csname gymnastics}}. (github issue 669)

Undo math alphabet allocations if necessary
\endtext{}X, or more exactly the 8-bit versions of \endtext{}X, such as pdf\endtext{}X, have a hard limit of 16 on the number of different math font groups \texttt{\fam} or \texttt{\mathgroup} that can be used in a single formula. For each symbol font declared (by a package or in the preamble) an extra math group is allocated, and the same happens for each math alphabet, (such as \texttt{\mathbf}) once it gets used anywhere in the document. Up to now, these math alphabet allocations were permanent, even if they were used only once; the result was that in complex documents you could easily run out of available math font groups. The only remedy for this was to define your own math version, which is a complicated and cumbersome process.

This situation has now been improved by the introduction of a new counter \localmathalphabets: this counter governs how many of the math group slots are assigned locally when a new math alphabet (and a new math group) is needed. Once the current formula is finished, every such further (local) allocation is undone, giving you a fighting chance of being able to use different new math alphabets in the next formula.

The default value of \localmathalphabets{} is 2, but if you need more local alphabets because of the complexity of your document, you can set this to a higher value such as 4 or 5. Setting it even higher is possible, but this would seldom be useful because many group slots will be taken up by symbol fonts and such slots are always permanently allocated, whether used or not. (github issue 676)

New default value for \tracinglostchars
In 2021 all \endtext{}X engines were enhanced so that \tracinglostchars supported the value 3 to turn missing characters into errors and not just warnings. This engine change made us realize that \endtext{}X should set a better default value for this parameter (previously, the warning was written only to the transcript file). Using the now available value of 3 as the default would be ideal, but for compatibility reasons we have only increased it to 2 in the kernel. However, we recommend setting \tracinglostchars=3, in either a package or the preamble of your documents: this is because having missing glyphs in the output is definitely an error and should therefore be flagged as such (to ensure that it gets proper attention). Further reasons, related especially to Unicode engines, for making this recommended change are explained later in this newsletter (in connection with the misuse of text accents in math mode).

\PackageNote and \ClassNote added
\endtext{}X offers these three commands: \PackageError to signal errors that stop the processing; \PackageWarning to generate a warning message on the terminal but continue with the processing; and \PackageInfo to provide some information that is only written to the \texttt{.log} file but not sent to the terminal. What has not existed up to now is a way to provide information on the terminal that identifies itself as coming from a specific package but which does not claim to be a warning. (Packages that wanted to write to the terminal used \PackageWarning even though the information was not in fact a warning.)

We have therefore now added \PackageNote (and the closely related \PackageNoteNoLine); these identify themselves as “informational”, but they still go to the terminal and not only to the \texttt{.log} file. Similar commands exist for classes and so there too we have new commands: \ClassNote and \ClassNoteNoLine. (github issue 613)
New \ShowFloat command
The package \texttt{ftrace} offers a (fairly low-level but very detailed) way to trace \LaTeX{}'s float mechanism. This can help in understanding why a certain float is placed into a certain region, or why it shows up unexpectedly on a later page. \LaTeX{} stores floats in registers named \texttt{bx@A}, \texttt{bx@B}, etc., and these names show up in the tracing information.

To display the contents of a float register, you can now say \texttt{\ShowFloat{identifier}} where \texttt{identifier} is the uppercase letter (or letters) after \texttt{bx@} in the register name shown in the tracing. If additional registers have been allocated (with \texttt{extrafloats}), the \texttt{identifier} can also be a number. The command is generally available, whether or not you have loaded \texttt{ftrace}, because it is also useful when interpreting the tracing output of the \texttt{fewerfloatpages} package.

New argument for \counterwithin/without
The commands \counterwithout and \counterwithin each now has an additional optional argument, similar to that of the command \numberwithin from \texttt{amsmath}, for which these are now the preferred replacements. This optional argument specifies the format of the counter, such as \texttt{roman}; the default value is \texttt{arabic}. Alternatively, you can use a starred form, in which case the format of the counter is not altered at all.

Tests for package and class loading
To test whether a package has been loaded you can now use \texttt{\IfPackageLoadedTF {(package)} {(true)}{(false)}} and, based on the result, execute different code. It is also possible to check whether the package was loaded with certain options. This is done with \texttt{\IfPackageLoadedWithOptionsTF}. It takes four arguments: \{(package)\} \{(option-list)\} \{(true)\} \{(false)\}. It uses the \texttt{(false)} code if one or more options in the \texttt{(option-list)} were not specified when loading the package, or if the package has never been loaded. Both commands can be used anywhere in the document, i.e., they are not restricted to the preamble.\footnote{This is now also true for the corresponding internal commands, e.g., \texttt{\IfPackageLoaded}, that had this restriction in the past.}

For classes, similar commands, with \texttt{Package} replaced by \texttt{Class} in the name, are provided. \texttt{(github issue 621)}

Better handling for a misuse of \texttt{\include}
The command \texttt{\include} has by now been used quite often, but erroneously, to input a variety of files in the preamble of the document (before \texttt{\begin{document}}). Therefore \LaTeX{} now warns about such bad use of \texttt{\include}. As a recovery action it will nevertheless input the specified file if it exists (this is as before). Note, however, that this is now done without any adjustments to the \texttt{.aux} file settings and without running the \texttt{\include} file hooks (only the generic file hooks from \texttt{\InputIfFileExists} are run). \texttt{(github issue 645)}

Code improvements
Use OpenType version of Latin Modern Upright Italic font
When a Latin Modern font is used with the TU encoding under \texttt{Xe\LaTeX} or \texttt{Lua\LaTeX} and fontshape \texttt{ui} is requested, \LaTeX{} now uses the OpenType version of the font instead of substituting the \texttt{(T1-encoded) Type 1 version}.

Additional Extended Latin characters predefined
More characters, such as \texttt{k} (U+1E31), are now predefined and do not need a \texttt{\DeclareUnicodeCharacter} declaration. \texttt{(github issue 593)}

Check \texttt{endfoo} in \texttt{NewDocumentEnvironment}
The \texttt{newenvironment} command has always checked that neither \texttt{foo} nor \texttt{endfoo} exists before creating a \texttt{foo} environment. In contrast (for historical reasons) the more recently introduced command \texttt{NewDocumentEnvironment} checked only for \texttt{foo}. The behavior of \texttt{NewDocumentEnvironment} now aligns with that of \texttt{newenvironment}, except that it gives distinct errors concerning the existence of \texttt{foo} and \texttt{endfoo}.

Improve the error message \texttt{begin ended by ...}
In the past it was possible to get an error message along the lines of \texttt{\begin{foo} ended by \end{foo}”. This could happen when the environment name was partly hidden inside a macro. It happened because the test was comparing the literal strings, whereas in the error message these got fully expanded. This has now been changed to show a more sensible error message. \texttt{(github issue 587)}

Pick up all arguments to \texttt{\contentsline}
A \texttt{\contentsline} command in the \texttt{.toc} file is always followed by four arguments, the last one being empty except when using the \texttt{hyperref} package. The \texttt{\contentsline} command itself only used the first three arguments and it relied on the fourth being empty (and thus doing no harm). But this assumption is not always correct: e.g., if you at first decide to load \texttt{hyperref} but then later you remove this loading from the preamble. So now all four arguments are picked up, with the fourth being saved away so that it can be used by \texttt{hyperref}. \texttt{(github issue 633)}

Allow dropping a math list in Lua\LaTeX{} callback
The \texttt{Lua\LaTeX} callbacks \texttt{pre_mlist_to_hlist_filter} and \texttt{post_mlist_to_hlist_filter} no longer create an error when the callback handler indicates removal of the entire math list. \texttt{(github issue 644)}
Extended label handling in package code

Since 2020, as noted in \texttt{\LaTeX} News 32 \cite{n32}, \LaTeX{} has recorded the name of the counter associated with the current label in the internal command \verb|\@currentcounter|. This facility (originally from the \texttt{zref} package of Heiko Oberdiek) can be used to generate prefixes such as “Figure” before the reference text, as long as the counter is not counting different objects in a single sequence (e.g., lemmas and theorems). In the most common cases the current label is set by \verb|\refstepcounter|, which automatically stores the counter name; but some constructs (alignments and footnotes) may need to store the current label directly and so for these it is useful to update additionally \verb|\@currentcounter| so as to store this counter name.

In this release both the footnote command in the kernel and also some of the environments in the \texttt{amsmath} package have been updated in this way. We encourage the maintainers of any class or package files that define \verb|\@currentlabel| to also set \verb|\@currentcounter| at the same point. \hfill (github issue 300, 687)

Better message if text accent used in math mode

Using text accents like \textbackslash{}^{} in math does not work (and \TeX{} explicitly provides math accents such as \textbackslash{}hat for accessing such symbols in math mode). Therefore \LaTeX{} issued a warning when such a wrongly placed accent was encountered and this was often followed by a strange, and apparently unrelated, low-level error. This has now been changed so that the message from this error is at least about accents, which we hope is less puzzling.

Discussion of such warnings or errors reminds us to reinforce here a recommendation from earlier in this newsletter, as part of the item on the value of \texttt{tracinglostchars}. Using \TeX{} implementations from 2020 onwards, any warning that concerns missing characters can be converted to an error by setting \texttt{tracinglostchars} to 3; we therefore now recommend changing this setting to 3, especially for Unicode engines where such missing characters are common (because no font supports the full Unicode range). \hfill (github issue 643)

Bug fixes

Replicate argument processors for all embellishments in command declarations

There was a bug in \texttt{ltcmd} (formerly \texttt{xparsel}) that caused commands to misbehave if they were defined with embellishments and argument processors. In that case, only one (possibly void) argument processor would be added to the full set of embellishment arguments, resulting in too few processors in some cases and thus leading to unpredictable behavior. This bug has been fixed by applying the same argument processors to all the embellishments in a set, so that a declaration like:\texttt{\NewDocumentCommand\foo}{>{\TrimSpaces}e{\^}}
{(#1)[#2]} \foo^-\{ a \}_{ b }\}
will now correctly apply \texttt{\TrimSpaces} to both arguments. \hfill (github issue 639)

Correct case changing of \texttt{\ij} and \texttt{\IJ}

The ligatures “\texttt{ij}” and “\texttt{IJ}”, as used in Dutch, are available for (most \texttt{\LaTeX} fonts) only when the commands \texttt{\ij} or \texttt{\IJ} are used, or when you enter them as the Unicode characters U+0133 or U+0132. However, when using \texttt{OT1} or \texttt{T1} encoded fonts in \texttt{pdf\LaTeX}, the upper or lower casing with \texttt{\MakeUppercase} and \texttt{\MakeLowercase} would always fail regardless of the input method. This has now been corrected. At the same time we improved the hyphenation results for words containing this ligature (when using the \texttt{OT1} encoding). \hfill (github issue 658)

Legacy font series default changes

In the past, changes to the font series defaults were made by directly altering \texttt{\bfdefault} or \texttt{\mddefault}. Since 2020 there is now \texttt{\DeclareFontSeriesDefault} that allows more granular control: with this declaration you can alter the default for individual meta font families by, for example, changing the bold setting only for the sans serif family, without changing it for \texttt{\rmfamily} or \texttt{\ttfamily}. See \cite{texdoc3} for more details.

For backwards compatibility, changing \texttt{\bfdefault} with \texttt{\bfdefault} remained possible; if used, this alters the setting for all meta families in one go. This alteration cannot be done when the \texttt{\bfdefault} happens and it was therefore delayed until the next time \texttt{\bfseries} or \texttt{\mdseries} was executed. However, the problem with that approach was that any call to \texttt{\bfdefault} in the meantime was overwritten; thus, these two approaches didn’t work well in combination. There was a problem because older font packages use the legacy method while newer ones use \texttt{\bfdefault}.

This has now been resolved by changing \texttt{\bfdefault} to do any necessary resetting prior to setting the new defaults. \hfill (github issue 663)

Use of \texttt{#} in \texttt{\textbf} and similar commands

Previously you could not use the macro parameter character \texttt{#} in inline functions within the argument of \texttt{\textbf} or similar text font commands. An internal definition is now guarded with \texttt{\unexpanded} so that the use of \texttt{#} here no longer generates an error. \hfill (github issue 665)

Changes to packages in the \texttt{amsmath} category

Improved compatibility with \texttt{hyperref}

This change in \texttt{amsmath} fixes a spacing problem caused by the method used in \texttt{hyperref} to change the equation
environment. For simplicity, an explicit, low-level (hence possibly temporary) patch has been added to \texttt{amsmath}: this consists of an extra, empty (hence invisible) \texttt{\mathopen} atom (with no mathematical meaning) at the start of the environment’s mathematical content. (\texttt{github issue 652})

**Changes to packages in the graphics category**

\texttt{graphicx}: New key, for alt text
A new key, \texttt{alt}, has been added to \texttt{\includegraphics} to support the addition of descriptive text that is important for accessibility. This key is unused by default; it can be deployed by extension packages and it will provide useful support for other future possibilities. (\texttt{github issue 651})

**Changes to packages in the tools category**

\texttt{array}: No \texttt{\mathsurround} around a \texttt{tabular}
A \texttt{tabular} environment is typeset (internally) as an \texttt{array} environment with special settings, and it therefore uses (hidden) math mode. Since it is not in fact a math formula, no extra space from \texttt{\mathsurround} should be added (the spacing around the \texttt{tabular} should not get changed). Note that this bug has been present “forever”, which shows that \texttt{\mathsurround} is never used, or at least its use is never noticed. At any rate, this bug has now finally been fixed. (\texttt{github issue 614})

\texttt{longtable}: Improvements after a section heading
The \texttt{longtable} environment now sets the \texttt{\@nobreakfalse} flag to correct the typesetting when a table immediately follows a heading. Previously the spacing and indentation changes that are required immediately after a section heading were incorrectly triggered within the next paragraph (if any) following the table. A similar test for \texttt{if\@noskipsec} has been added, so that a table is correctly placed after a run-in heading rather than appearing before that heading. (\texttt{github issues 131 and 173})

\texttt{multicol}: Better column break control
From version 1.9 onwards \texttt{\columnbreak} accepts an optional argument (like \texttt{\pagebreak}) in which you can specify the desirability of breaking the column after the current line: supported values are 0 to 4, with higher numbers indicating increased desirability. This version also adds \texttt{\newcolumn}, which forces a break but runs the column short (comparable to \texttt{\newpage} for pages). (\texttt{github issue 682})

\texttt{varioref}: Improved handling of missing labels
If an undefined label is referenced, \texttt{varioref} makes a default definition so that later processing finds the right structure (two brace groups inside \texttt{\textbackslash r\textbackslash @\{label\}}) However, if \texttt{nameref} or \texttt{hyperref} is loaded, this data structure changes to having five arguments; this could cause low-level errors in some cases. The code has therefore now been changed to avoid these errors. (\texttt{https://tex.stackexchange.com/q/603948})

**References**


[2] \LaTeX\ documentation on the \LaTeX\ Project Website. https://latex-project.org/help/documentation/


[4] \LaTeX\ Project Team: \LaTeX\ 2ε news 32. https://latex-project.org/news/latex2e-news/ltnews32.pdf

[5] \LaTeX\ Project Team: \LaTeX\ 2ε news 33. https://latex-project.org/news/latex2e-news/ltnews33.pdf
Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>75</td>
</tr>
<tr>
<td>Document metadata interface</td>
<td>75</td>
</tr>
<tr>
<td>The latex-lab bundle</td>
<td>76</td>
</tr>
<tr>
<td>A new mark mechanism for \LaTeX</td>
<td>76</td>
</tr>
<tr>
<td>A key/value mechanism for option handling</td>
<td>77</td>
</tr>
<tr>
<td>New or improved commands</td>
<td>77</td>
</tr>
<tr>
<td>Floating point and integer calculations</td>
<td>77</td>
</tr>
<tr>
<td>CamelCase commands for changing arguments to csnames</td>
<td>77</td>
</tr>
<tr>
<td>Testing for (nearly) empty arguments</td>
<td>78</td>
</tr>
<tr>
<td>Better allocator for Lua command ids</td>
<td>78</td>
</tr>
<tr>
<td>Starred command version for \ref, \Ref and \pageref</td>
<td>78</td>
</tr>
<tr>
<td>Preparation for supporting PDF in backends</td>
<td>78</td>
</tr>
<tr>
<td>Code improvements</td>
<td>78</td>
</tr>
<tr>
<td>Protect\ed UTF-8 character definitions</td>
<td>78</td>
</tr>
<tr>
<td>A small update to \obeylines and \obeyspaces</td>
<td>78</td>
</tr>
<tr>
<td>doc upgraded to version 3</td>
<td>79</td>
</tr>
<tr>
<td>doc can now show dates in change log</td>
<td>79</td>
</tr>
<tr>
<td>\ltxdoc gets options ncfg and doc2</td>
<td>79</td>
</tr>
<tr>
<td>Lua\TeX\ callback improvements</td>
<td>79</td>
</tr>
<tr>
<td>Class proc supports twoside</td>
<td>79</td>
</tr>
<tr>
<td>Croatian character support</td>
<td>79</td>
</tr>
<tr>
<td>Cleanup of the Unicode declaration interface</td>
<td>79</td>
</tr>
<tr>
<td>New hook: include/excluded</td>
<td>80</td>
</tr>
<tr>
<td>Input support for normalized angle brackets</td>
<td>80</td>
</tr>
<tr>
<td>Bug fixes</td>
<td>80</td>
</tr>
<tr>
<td>Using \DeclareUnicodeCharacter with C1 control points</td>
<td>80</td>
</tr>
<tr>
<td>Fix \ShowCommand when used with \ltxcmd</td>
<td>80</td>
</tr>
<tr>
<td>Make \cite{} produce a warning</td>
<td>80</td>
</tr>
<tr>
<td>Fix adding cmd hooks to simple macros</td>
<td>80</td>
</tr>
<tr>
<td>Warn if shipout/lastpage hook is executed too early</td>
<td>80</td>
</tr>
<tr>
<td>More consistent use of cramped math styles in \ltx\TeX</td>
<td>80</td>
</tr>
<tr>
<td>Fixed bug when setting hook rules for one-time hooks</td>
<td>80</td>
</tr>
</tbody>
</table>

Changes to packages in the amsmath category 81
- amsopn: Do not reset \operator@font 81
- amsmath: Error in \shoveleft 81
- amsmath and amsopn: Robustify user commands 81

Changes to packages in the graphics category 81
- Color in formulas 81
- Fix locating files with \graphicspath 81

Changes to packages in the tools category 81
- multicol: Fix \newcolumn 81
- bm: Fix for amsmath operators 81

Introduction

The 2022 June release of \LaTeX\ is again focusing on improvements made for our multi-year project to automatically offer tagged PDF output [1]. These are the new document metadata interface, the new mark mechanism for \LaTeX\, a standard key/value approach for options, and the introduction of the latex-lab area for temporary code that can be optionally loaded by a document (when \DocumentMetadata is used with certain test keys). These additions are described in the first sections. Related to this effort there are updates to hyperref and tagpdf, both of which have their own distributions.

As usual, we also added a number of smaller improvements and bug fixes in various components of \LaTeX. Perhaps the most interesting ones (for some users) are direct support for floating point arithmetic (via \fpeval; see below) and the ability to properly color parts of math formulas without introducing spacing problems. For this we now offer the command \mathcolor; see the description near the end of the newsletter. There is also a new major release of the doc package that supports a more fine-grained classification of code elements and properly supports hyperref.

Document metadata interface

Until recently there was no dedicated location to declare settings that affect a document as a whole. Settings had to be placed somewhere in the preamble or as class options or sometimes even as package options. For some such settings this may cause issues, e.g., setting the PDF version is only possible as long as the PDF output file has not yet been opened which can be caused by loading one or the other package.
For the “\LaTeX Tagged PDF project” [1] further metadata about the whole document (and its processing) need to be specified and again this data should be all placed in a single well-defined place.

For this reason we introduce the new command \DocumentMetadata to unify all such settings in one place. The command expects a key/value list that describes all document metadata for the current document. It is only allowed to be used at the very beginning of the document, i.e., the declaration has to be placed before \documentclass and will issue an error if found later.

At this point in time we provide only the bare command in the format; the actual processing of the key/value is defined externally and the necessary code will be loaded if the command is used. This scheme is chosen for two reasons: by adding the command in the kernel it is available to everybody without the need to load a special package using \RequirePackage. The actual processing, though, is external so that we can easily extend the code (e.g., offering additional keys or changing the internal processing) while the above-mentioned project is progressing. Both together allows users to immediately benefit from intermediate results produced as part of the project, as well as offering the \LaTeX Project Team the flexibility to enable such intermediate results (for test purposes or even production use) in-between and independently of regular \LaTeX releases. Over time, tested and approved functionality can then seamlessly move into the kernel at a later stage without any alterations to documents already using it. At the same time, not using the new consolidated interface means that existing documents are in no way affected by the work that is carried out and is in a wider alpha or beta test phase.

Documentation about the new command and already existing keys are in \texttt{ltmeta} (part of source2e.pdf) and \texttt{documentmetadata-support.pdf} and also in the documentation of the \texttt{pdfmanagement-testphase} package.

Package and class authors can test if a user has used \DocumentMetadata with \IfDocumentMetadataTF.

The \texttt{latex-lab} bundle

We added a new \texttt{latex-laboratory} bundle in which we place new code that is going to be available only through a \DocumentMetadata declaration and that is—most importantly—work under development and subject to change without further notice. This means that commands and interfaces provided there may get altered or removed again after some public testing. The code can be accessed through the \DocumentMetadata key \texttt{testphase}. Currently supported values are \texttt{phase-I} and \texttt{phase-II} that enable code of the tagged PDF project (phase I is frozen, and phase II is the phase we are currently working on). With

\DocumentMetadata{testphase=phase-II}

you currently enable tagging for paragraphs and footnotes; more document elements will follow soon.

Eventually, code will move (once considered stable) from the testphase into the \LaTeX kernel itself. Tagging will continue to require a \DocumentMetadata declaration, but you will then be able to drop the \texttt{testphase} key setting.

A new mark mechanism for \LaTeX

The mark mechanism is \TeX’s way to pass information to the page-building process, which happens asynchronously, in order to communicate relevant data for running headers and footers to the latter, e.g., what is the first section on the page or the last subsection, etc. However, marks may also be used for other purposes. The new kernel module provides a generalized mechanism for marks of independent classes.

The \TeX engines offer a low-level mark mechanism to communicate information about the content of the current page to the asynchronous operating output routine. It works by placing \mark commands into the source document.

This mechanism works well for simple formats (such as plain \TeX) whose output routines are only called to generate pages. It fails, however, in \LaTeX (and other more complex formats), because here the output routine is sometimes called without producing a page, e.g., when encountering a float and placing it into one of the float regions. When that happens \TeX’s \texttt{topmark} no longer reflects the situation at the top of the next page when that page is finally boxed.

Furthermore, \TeX only offered a single mark while \LaTeX wanted to keep track of more than one piece of information. For that reason, \LaTeX implemented its own mark mechanism where the marks always contained two parts with their own interfaces: \texttt{markboth} and \texttt{markright} to set marks and \texttt{leftmark} and \texttt{rightmark} to retrieve them.

Unfortunately, this extended mechanism, while supporting scenarios such as chapter/section marks, was far from general. The mark situation at the top of a page (i.e., \texttt{topmark}) remained unusable and the two marks offered were not really independent of each other because \texttt{markboth} (as the name indicates) was always setting both.

The new mechanism now available in \LaTeX starting with the 2022 release overcomes both issues:

- It provides arbitrary many, fully independent named marks, that can be allocated and from that point onwards used.
• It offers access for each such mark to retrieve its top, first, and bottom value separately.
• Furthermore, the mechanism is augmented to give access to marks in different “regions”, which may be other than full pages.

The legacy interfaces, e.g., \markboth, are kept. Thus classes and packages making use of them continue to work seamlessly. To make use of the extended possibility a new set of commands for the declaration of mark classes, setting their values and querying their state (in the output routine) is now available in addition. You find the documentation for the new interfaces together with examples and further notes on the mechanism in the file ltmarks-doc.pdf. Just call texdoc ltmarks-doc to display it on your computer.

A key/value approach to option handling
The classical \LaTeX{} method for handling options, using \ProcessOptions, treats each entry as a string. Many package authors have sought to extend this handling by treating each entry as a key–value pair (keyval) instead. To date, this has required the use of additional packages, for example kvoptions.

The \LaTeX{} team have for some time offered the package l3keys2e to allow keyvals defined using the L3 programming layer module l3keys to act as package options. This ability has now been integrated directly into the kernel. As part of this integration, the syntax for processing keyval options has been refined, such that \ProcessKeyOptions has been added. This \ProcessKeyOptions will now automatically pick up the package name as the key \texttt{family}, unless explicitly given as an optional argument:

\ProcessKeyOptions[family]

To support creating key options for this mechanism, the new command \DeclareKeys has been added. This works using the same general approach as l3keys or pgfkeys: each key has one or more properties which define its behavior.

Options for packages which use this new approach will not be checked for clashes by the kernel. Instead, each time a \usepackage or \RequirePackage line is encountered, the list of options given will be passed to \ProcessKeyOptions. Options which can only be given the first time a package is loaded can be marked using the property \texttt{usage = load}, and will result in a warning if used in a subsequent package loading line.

Package options defined in this way can also be set within a package using the new command \SetKeys, which again takes an optional argument to specify the \texttt{family}, plus a mandatory one for the options themselves.

\textbf{New or improved commands}

\textit{Floating point and integer calculations}

The L3 programming layer offers expandable commands for calculating floating point and integer values, but so far these functions have only been available to programmers, because they require \ExplSyntaxOn to be in force. To make them easily available at the document level, the small package xfp defined \fpeval and \inteval.

An example of use could be the following:

\LaTeX{} can now compute:
\begin{verbatim}
\[ \frac{\sin (3.5)}{2} + 2\cdot 10^{-3} = \fpeval{\sin(3.5)/2 + 2e-3} \]
\end{verbatim}

which produces the following output:

\LaTeX{} can now compute:
\[
\frac{\sin(3.5)}{2} + 2 \cdot 10^{-3} = -0.173391613848099
\]

These two commands have now been moved into the kernel and in addition we also provide \dimeval and \skipeval. The details of their syntax are described in usrguide3.pdf. The command \fpeval offers a rich syntax allowing for extensive calculations, whereas the other three commands are essentially thin wrappers for \numexpr, \dimexpr, and \glueexpr—therefore inheriting some syntax peculiarities and limitations in expressiveness.

\texttt{\newcommand\calculateheight[1]{{\setlength\textheight{\dimeval{\topskip + \baselineskip * \inteval{#1-1}}}}}}

The above, for example, calculates the appropriate \texttt{textheight} for a given number of text lines.

\texttt{(github issue 711)}

\textit{CamelCase commands for changing arguments to csnames}

It is sometimes helpful to “construct” a command name on the fly rather than providing it as a single \texttt{...} token. For these kinds of tasks the \LaTeX{}3 programming layer offers a general mechanism (in the form of \exp_args:N... and \cs_generate_variant:Nn). However, when declaring new document-level commands with \NewDocumentCommand or \NewCommandCopy, etc., the L3 programming layer may not be active, and even if it is, mixing CamelCase syntax with L3 programming syntax is not really a good approach. We have therefore added the commands \UseName and \ExpandArgs to assist in such situations, e.g.,

\begin{verbatim}
\NewDocumentCommand\newcopyedit{mO{red}}{\exp_args:N...\ExpandArgs{c}\NewDocumentCommand{#1}{s m}{\stepcounter{todo#1}}\IfBooleanTF {##1}}
\end{verbatim}
\newcommand{\greeting}{world}

\begin{quote}
\textit{Testing for (nearly) empty arguments}

In addition to \texttt{\IfNoValueTF} to test if an optional argument was provided or not, there is now also \texttt{\IfBlankTF}, which tests if the argument is empty or contains only blanks. Based on the result it selects a true or false code branch. As usual, the variants \texttt{\IfBlankT} and \texttt{\IfBlankF} are also provided for use when only one branch leads to some action. Further details and examples are given in \html{usrguide3.pdf}

\textit{Better allocator for Lua command ids}

In \LaTeX{} we already had the \texttt{\newluafunction} macro which allocates a Lua function identifier which can be used to define commands with \texttt{\luadef}. But this always required two steps: \texttt{\newluafunction} defines the passed control sequence as an integer, which then has to be used to define the actual Lua command with \texttt{\luadef}. After that, the integer is no longer needed. This was inconsistent with other allocators. Therefore we added two new allocators \texttt{\newluacmd} and \texttt{\newexpandableluacmd} which directly define a control sequence invoking the allocated Lua function. The former defines a non-expandable Lua command, the latter an expandable one. Of course, the associated Lua function still has to be defined by assigning a function to the \texttt{\lua.get_functions_table()} table. The required index is available in \texttt{\allocationnumber}.

An example could be
\begin{verbatim}
\newluacmd \greeting
\directlua{
\lua.get_functions_table()
  [tex.count.allocationnumber] = function()
    local name = token.scan_argument()
    tex.sprint('Hello ', name, '!')
  end
}
\end{verbatim}

\textit{Starred command version for \texttt{\ref}, \texttt{\Ref} and \texttt{\pageref}}

For a long time \texttt{hyperref} has provided starred versions for the reference commands which do not create active links. This syntax extension required users and package authors to check if \texttt{hyperref} was loaded and adjust the coding accordingly or take the starred forms out if text was copied to a document without \texttt{hyperref}. The commands have now been aligned with the \texttt{hyperref} usage and always allow an optional star. The \texttt{showkeys} package has been updated to handle the starred versions too, both with \texttt{hyperref} or \texttt{nameref} and without. The commands are defined with \texttt{\NewDocumentCommand} and so no longer expand when written to auxiliary files. This reduces the number of compilations needed to resolve references in captions and sectioning commands. The package \texttt{ifthen} has been updated to ensure that \texttt{\pageref} can still be used inside tests like \texttt{\isodd}.

\textit{Preparation for supporting PDF in backends}

At the current point in time, basic support for PDF in backends is not part of \LaTeX{} core; it is provided by an external package like \texttt{hyperref}. At some time in the future that work will be placed into the kernel but for now it is separate and has to be explicitly loaded in the document. To enable class and package authors to support PDF-specific tasks like the creation of link targets without having to test first if \texttt{hyperref} has been loaded, dummy versions of the commands \texttt{\MakeLinkTarget}, \texttt{\LinkTargetOn}, \texttt{\LinkTargetOff} and \texttt{\NextLinkTarget} are provided.

\textit{Code improvements}

\texttt{\protected \UTF-8 character definitions}

The characters defined via \texttt{utf8.def} are now defined as \texttt{\protected} macros. This makes them safe to use in expansion contexts where the classic \texttt{\protect} mechanism is not enabled, notably L3 programming layer \texttt{e} and \texttt{x} arguments.

Related to this change \texttt{\MakeUppercase} and \texttt{\MakeLowercase} have been updated to use the Unicode-aware case changing functions \texttt{\textcase:} in place of the \TeX{} primitive \texttt{\lowercase}. The \texttt{\NoCaseChange} command from the \texttt{textcase} package has also been added.

Note: for technical reasons these low-level character handling changes will not be rolled back if the format version is rolled back using the \texttt{latexrelease} package rollback mechanism.

\textit{A small update to \texttt{\obeylines} and \texttt{\obeyspaces}}

The plain \TeX{} versions of \texttt{\obeylines} and \texttt{\obeyspaces} make \texttt{\^M} and \texttt{\_} active and force them to execute \texttt{\par}
and \space, respectively. Don Knuth makes a remark in the \TeXbook that one can then use a trick such as
\let\par=\cr \obeylines \halign{...

However, redefining \par like this may lead to all kinds of problems in \LaTeX. We have therefore changed the commands to use an indirect: the active characters now execute \obeyedline and \obeyedspace, which in turn do what the hardwired solution did before.

This • means • that • it • is • now • possible • to
• achieve • special • effects • in • a • safe • way.
• This • paragraph, • for • example, • was •
produced • by • making • \obeyedspace • generate
• \{\textbullet\} and • enabling •
\obeyespaces • within • a • quote • environment.

Thus, if you are keen to use the plain \TeX trick, you need to say \let\obeyedline=\cr now. (github issue 367)

doc upgraded to version 3
After roughly three decades the doc package received a cautious uplift, as already announced at the 2019 TUG conference—changes to doc are obviously always done in a leisurely manner.

Given that most documentation is nowadays viewed on screen, hyperref support is added and by default enabled (suppress it with option nohyperref or alternatively with hyperref=false) so the internal cross-references are properly resolved including those from the index back into the document.

Furthermore, doc now has a general mechanism to define additional “doc” elements besides the two \Macro and \Env it has known in the past. This enables better documentation because you can now clearly mark different types of objects instead of simply calling them all “macros”. If desired, they can be collected together under a heading in the index so that you have a section just with your document interface commands, or with all parameters, or ...

The code borrows ideas from Didier Verna’s doc package (although the document level interface is different) and it makes use of Heiko Oberdiek’s hypdoc package, which at some point in the future will be completely integrated, given that its whole purpose it to patch doc’s internal commands to make them hyperref-aware.

All changes are expected to be upward compatible, but if you run into issues with older documentation using doc a simple and quick solution is to load the package as follows: \usepackage[doc][v2]

doc can now show dates in change log
Up to now the change log was always sorted by version numbers (ignoring the date that was given in the \changes command). It can now be sorted by both version and date if you specify the option reportchangedates on package level and in that case the changes are displayed with

\〈version⟩ − \〈date⟩

as the heading (instead of just \〈version⟩), when using \PrintChanges. (github issue 531)

\ltxdoc gets options nocfg and doc2
The \LaTeX sources are formatted with the \ltxdoc class, which supports loading a local config file \ltxdoc.cfg. In the past the \LaTeX sources used such a file but it was not distributed. As a result reprocessing the \LaTeX sources elsewhere showed formatting changes. We now distribute this file which means that it is loaded by default. With the option nocfg this can be prevented.

We also added a doc2 option to the class so that it is possible to run old documentation with doc version 2, if necessary.

\LuaTeX callback improvements
The \LuaTeX callbacks \hpack_quality and \vpack_quality are now exclusive and therefore only allow a single handler. The previous type list resulted in incorrect parameters when multiple handlers were set; therefore, this only makes an existing restriction more explicit.

Additionally the return value true for list callbacks is now handled internally and no longer passed on to the engine. This simplifies the handling of these callbacks and makes it easier to provide consistent interfaces for user-defined list callbacks.

Class proc supports twoside
The document class proc, which is a small variation on the \article class, now supports the \twoside option, displaying different data in the footer line on recto and verso pages. (github issue 704)

Croatian character support
The default inputenc support has been extended to support the 9 characters ĐŽ, Đž, đž, LJ, Lj, lj, NJ, Nj, nj, input as single UTF-8 code points in the range U+01C4 to U+01CC. (github issue 723)

Cleanup of the Unicode declaration interface
When declaring encoding specific commands for the Unicode (TU) encoding some declarations (e.g., \DeclareUnicodeComposite) do not have an explicit argument for the encoding name, but instead use the command \UnicodeEncodingName internally. There was one exception though: \DeclareUnicodeAccent required an explicit encoding argument. This inconsistency has now been removed and the encoding name is always implicit. To avoid a breaking change for a few packages on CTAN, \DeclareUnicodeAccent still accepts three arguments if the second argument is TU or
\UnicodeEncodingName. Once all packages have been updated this code branch will get removed.

At the same time we added \DeclareUnicodeCommand and \DeclareUnicodeSymbol for consistency. They also use \UnicodeEncodingName internally, instead of requiring an encoding argument as their general purpose counterparts do. (github issue 253)

**New hook: include/excluded**

A few releases ago we introduced a number of file hooks for different types of files; see [2] and in particular [4]. The hooks for \include files now have an addition: if such a file is not included (because \includeonly is used and its \langle{name} is not listed in the argument) then the hooks include/excluded and include/{name}/excluded are executed in that order—of course, only if they contain code. This happens after \LaTeX has loaded the .aux file for this include file, i.e., after \LaTeX has updated its counters to pretend that the file was seen.

**Input support for normalized angle brackets**

Source files containing \langle or \rangle directly written as Unicode codepoints U+2329 and U+232A no longer break when the source file gets normalized under Unicode normalization rules. (github issue 714)

**Bug fixes**

**Using \DeclareUnicodeCharacter with C1 control points**

An error in the UTF-8 handling for non-Unicode \LaTeX has prevented \DeclareUnicodeCharacter being used with characters in the range hex 80 to 9F. This has been corrected in this release. (github issue 730)

**Fix \ShowCommand when used with ltcmd**

When \ShowCommand support was added for ltcmd in the previous release [3], a blunder in the code made it so that when \ShowCommand was used on a command defined with ltcmd, it only printed the meaning of the command in the terminal, but didn’t stop for interaction as it does elsewhere (mimicking \show). The issue is now fixed. (github issue 739)

**Make \cite{} produce a warning**

When the \cite command can’t resolve a citation label it issues a warning “Citation ‘(label)’ on page (page) undefined”. However, due to some implementation details a completely empty argument was always silently accepted. Given that there are probably people who write \cite{} with the intention to fill in the correct label later it is rather unfortunate if that is not generating a warning that something in the document is still amiss. This has finally been corrected and a warning is now generated also in this case. (github issue 790)

**Fix adding cmd hooks to simple macros**

A bug in how \LaTeX detected the type of a command caused a premature forced expansion of such commands, which, depending on their definition, could be harmless or could cause severe trouble. This has been fixed in the latest release. (github issue 795)

(https://tex.stackexchange.com/q/637565)

**Warn if shipout/lastpage hook is executed too early**

The hook shipout/lastpage is intended to place \specials into the last page shipped out. This is needed for some use cases, e.g., tagging. If that hook is nonempty and the user has added additional pages since the last run, then \LaTeX executes this hook too early, but until now without giving any indication that the document needs rerunning. This has now been corrected and an appropriate warning is given. (github issue 813)

**More consistent use of cramped math styles in \LaTeX**

Using Lua\LaTeX’s \Udelimiter to place a horizontally extensible glyph on top of a mathematical expression now causes the expression to be set in cramped style, as used in similar situations by traditional \TeX math rendering. Similarly, cramped style is now used for expressions set under such a delimiter using \Underdelimiter, but is no longer used when setting an expression on top of such extensible glyphs using \Uoverdelimiter. This new behavior follows \TeX’s rule that cramped style is used whenever something else appears above the expression. Additionally the math style of these constructs can now be detected using \mathstyle.

The old behavior can be restored by adding
\begin{verbatim}
\mathdefaultsmode=0
\end{verbatim}
to a document.

**Fixed bug when setting hook rules for one-time hooks**

If a \DeclareHookRule command is set for a one-time hook, it has to come before the hook gets used, because otherwise it never applies—after all, the hook is used only once. There was a bug in the implementation in that the sorting mechanism was still applied if the \DeclareHookRule declaration appeared while the one-time hook was executed, causing the spurious typesetting of the code labels and the hook name. This bug is now fixed and an error is raised when a new sorting rule is added to an already-used one-time hook.

A possible scenario in which this new error is raised is the following: package AAA declares a hook rule for \AtBeginDocument (i.e., \AtBeginDocument) to sort out the behavior between itself and some other package. Package BBB wants to load package AAA but only if it hasn’t been loaded in the preamble, so delays the loading to \AtBeginDocument. In that case the hook rule declared by AAA can no longer be applied and you get the error. If that happens the solution is to load the
Changes to packages in the amsmath category

\texttt{amsopn: Do not reset \operator@font}

The package \texttt{amsopn} used to define \texttt{\operator@font} but this command has been provided by the \LaTeX\ package for at least 14 years. As a result the definition in \texttt{amsopn} is equivalent to a reset to the kernel definition, which is unnecessary and surprising if you alter the math setup (e.g., by loading a package) and at a later stage add \texttt{amsmath}, which then undoes part of your setup. For this reason the definition was taken out and \texttt{amsmath/amsopn} now relies on the format definition.

In the unlikely event that you want the resetting to happen, use
\begin{verbatim}
\usepackage{amsmath}
\renewcommand{\operator@font}{\mathgroup\symoperators}
\end{verbatim}

\texttt{amsmath: Error in \shoveleft}

If \texttt{\shoveleft} started out with the words “plus” or “minus” it was misunderstood as part of a rubber length and led either to an error or was swallowed without trace. By adding a \texttt{\relax} this erroneous scanning into the argument of \texttt{\shoveleft} is now prevented.

\texttt{amsmath and amsopn: Robustify user commands}

Most user-level commands have been made robust in the \LaTeX\ kernel during the last years, but variant definitions in \texttt{amsmath} turned them back into fragile beings. We have now made most commands in \texttt{amsmath} and \texttt{amsopn} robust as well to match the kernel behavior. This also resolves a bug recently discovered in the \texttt{amsmath} package, which was due to \texttt{\big} not being robust after \texttt{amsmath} was loaded.

Changes to packages in the tools category

\texttt{multicol: Fix \newcolumn}

The recently added \texttt{\newcolumn} didn’t work properly if used in vertical mode, where it behaved like \texttt{\columnbreak}, i.e., spreading the column material out instead of running the column short.

\texttt{bm: Fix for amsmath operators}

An internal command used in the definition of operator commands such as \texttt{\sin} in \texttt{amsmath} has been guarded in \texttt{\bm} to prevent internal syntax errors due to premature expansion.

References

[1] Frank Mittelbach and Chris Rowley: \textit{\LaTeX\ Tagged PDF—A blueprint for a large project.}
https://latex-project.org/publications/indexbyyear/2020/

[2] \LaTeX\ Project Team: \textit{\LaTeX\ 2e news 32.}
[3] \LaTeX{} Project Team: \textit{\LaTeX{} 2e news 34}.
\url{https://latex-project.org/news/latex2e-news/ltnews34.pdf}

[4] Frank Mittelbach, Phelype Oleinik, \LaTeX{} Project Team: \textit{The ltfilehook documentation}.
Run \texttt{texdoc ltfilehook-doc} to view.
**Introduction**

The 2022-11 release of \LaTeX{} is largely a consolidation release where we made a number of minor improvements to fix some bugs or improve one or the other interface.

The only really important functionality that was added is described in the next section: the ability to easily define document-level commands and environments that accept a key/value list in one of its (usually optional) arguments, including the ability to determine if the argument does in fact contain such a key/value list or just a single “classical” value.

For the “Tagged \LaTeX{} Project” this functionality is very important because many document-level commands will need to accept such key/value lists, for example, to specify alternative text or overwrite default tagging if that becomes necessary in a document.

**Auto-detecting key/value arguments**

To allow extension of the core \LaTeX{} syntax, \texttt{ltcmd} now supports a \texttt{=} \ldots modifier when grabbing arguments. This modifier instructs \LaTeX{} that the argument should be passed to the underlying code as a set of key/values. If the argument does not “look like” a set of key/values, it will be converted into a single key/value pair, with the argument to \texttt{=} specifying the name of that key. For example, the \texttt{\caption} command could be defined as

\begin{verbatim}
\DeclareDocumentCommand\caption {s = {short-text} +O{#3} +m}
{...}
\end{verbatim}

which would mean that if the optional argument does \emph{not} contain key/value data, it will be converted to a single key/value pair with the key name \texttt{short-text}.

Arguments which begin with \texttt{=} are always interpreted as key/values even if they do not contain further \texttt{=} signs. Any \texttt{=} signs enclosed within $...$ or \texttt{(...)} \texttt{}, \texttt{..} \texttt{\ldots}$, i.e. in inline math mode, are ignored, meaning that only \texttt{=} outside of math mode will generally cause interpretation as key/value material.

In case the argument contains a “textual” \texttt{=} sign that is mistaken as a key/value indicator you can hide it using a brace group as you would do in other places, e.g.,

\begin{verbatim}
\caption[{Use of = signs}]
{Use of = signs in optional arguments}
\end{verbatim}

However, because \texttt{=} signs in math mode are already ignored, this should seldom be necessary.

**A note for font package developers**

**Encoding subsets for \texttt{TS1} encoded fonts**

The text companion encoding \texttt{TS1} is unfortunately not very faithfully supported in fonts that are not close cousins to the Computer Modern fonts. It was therefore necessary to provide the notion of “sub-encodings” on a per font basis. These sub-encodings are declared for a
font family with the help of a \DeclareEncodingSubset declaration, see [5] for details.

Maintainers of font bundles that include TS1 encoded font files should add an appropriate declaration into the corresponding ts1family.fd file, because otherwise the default subencoding is assumed, which is probably disabling too many glyphs that are actually available in the font.1

New or improved commands

Better language handling for case-changing commands

The commands \MakeUppercase, \MakeLowercase and \MakeTitlecase now automatically detect the locale currently in use when babel is loaded. This allows automatic adjustment of letter mappings where appropriate. They also accept a leading optional argument. This accepts a key–value list of control settings. At present, there is one key available: locale, which can also be accessed via the alias lang. This is intended to allow local setting of the language, which can be done using a BCP-47 descriptor. For example, this could be used to force Turkish case changing in otherwise English input

\MakeUppercase[lang = tr]{Ragıp Hulûsî Özdem}

yields RAGIP HULÜSİ ÖZDEM.

Code improvements

Support for slanted small caps in the EC fonts

For some time \LaTeX{} has supported the combination of the shapes small caps and italic/slanted. The EC fonts contain slanted small caps fonts but using them required the loading of an external package. Suitable font definitions have now been added to t1cmd.fd and so from now on

\usepackage[T1]{fontenc}
...
\textsc{\textit{Slanted Small Caps}};
\textsc{\textit{Italic Small Caps}};
\bfseries
\textsc{\textit{Bold Slanted Small Caps}};
\textsc{\textit{Bold Italic Small Caps}}.

will give the expected result: \textsc{Slanted Small Caps}; \textsc{Italic Small Caps}; \textbf{Bold Slanted Small Caps}; \textbf{Bold Italic Small Caps}.

Given that the Computer Modern fonts in T1 do not have real italic small caps but only slanted small caps, the latter is substituted for the former. This is why both work in the above, but there is no difference between the two (and you get a substitution warning for the \textit{\textsc{shape combination}). (github issue 782)

EC sans serif at small sizes

The EC (T1 encoded Computer Modern) sans serif fonts have errors at small sizes: the medium weight is bolder and wider than the bold extended. This makes them unusable at these small sizes. The default .fd file has therefore been adjusted to use a scaled down 8pt font instead. (github issue 879)

Improve font series handling with incorrect .fd files

By convention, the font series value is supposed to contain no m, unless you refer to the “medium” series (which is represented by a single m). For example, one should write c for “medium weight, condensed width” and not mc. This was one of the many space-conserving methods necessary in the early days of \LaTeX{}.

Some older .fd files do not obey that convention but use nc, bm, etc., in their declarations. As a result, some font selection scheme functionality was not working when confronted with such .fd files. We have therefore augmented \DeclareSymbolFont and \SetSymbolFont to strip any surplus m from their series argument so that they do not unnecessarily trigger font substitutions. Regardless of this support such .fd files should get fixed by their maintainers. (github issue 918)

Detect nested minipage environments

Nesting of minipage environments is only partially supported in \LaTeX{} and can lead to incorrect output, such as overfull boxes or footnotes appearing in the wrong place; see [1, p. 106]. However, until now there was no warning if that happened. This has been changed and the environment now warns if you nest it in another minipage environment that already contains footnotes. (github issue 168)

Robust commands in package options

With the standard key-based option handler added in the last release, or with contributed packages offering similar features, users may expect to be able to use a package option such as [font=\bfseries]. Previously this failed with internal errors as the option list was expanded via \edef. This has now been changed to use the existing command \protected@edef so that any \LaTeX{} robust command should be safe to pass to a key value option. (github issue 932)

Improve l3docstrip integration into docstrip

In 2020 we merged l3docstrip.tex into docstrip.tex to support the \%<@=\texttt{(module)}\> syntax of expl3; see [2]. However, this support was incomplete, because it didn’t cover docstrip lines of the form \%<+. . . > or \%<+. . . >. This was never noticed until now, because usually \%<+. . . > blocks are used. Now all lines in a .dtx file are subject to the @@ replacement approach. (github issue 903)
LuaTEX callback efficiency improvement

The mechanism for providing the \texttt{pre/post\_mlist\_to\_hlist\_filter} callbacks in LuaTEX has been improved to make it more reusable and to avoid overhead if these callbacks are not used.

\texttt{(github issue 830)}

Rule-based ordering for LuaTEX callback handlers

In LuaBTEX the callback handlers used to be called in the order in which they were registered in, but this was often rather fragile. It depends a lot on the load order and any attempts to enforce a different order required unregistering and reregistering the handlers to be reordered. Additionally, even if some ordering constraints where enforced that way, another package loaded later could accidentally overwrite it.

To improve this, we now order the callback handlers based on ordering rules similar to the hook rules.

When registering a callback which should run before or after another callback, \texttt{luatexbase.declare\_callback\_rule} can now be used to record this ordering constraint. For example

\texttt{luatexbase.add\_to\_callback('pre\_shaping\_filter', my\_handler, 'my\_name')}\texttt{luatexbase.declare\_callback\_rule('pre\_shaping\_filter', 'my\_name', 'before', 'other\_name')} will ensure that my\_handler will always be called before the handler registered as other\_name.

This also means that the order in which callbacks are registered no longer implicitly defines an order. Code which relied on this implicit order should now define the order rules explicitly.

Bug fixes

Prevent \TeX{} from losing a \texttt{\smash}

When \TeX{} is typesetting a fraction, it will rebox the material in either the numerator or denominator, depending on which is wider. If the repackaged part consists of a single box, that box gets new dimensions and if it was built using a \texttt{\smash} that effect vanishes (because a smash is nothing other than zeroing some box dimension, which now got undone). For example, in the line
\begin{verbatim}
\frac{1}{2} = \frac{1}{2^x} \neq \frac{100}{2^x}
\end{verbatim}

\texttt{(github issue 517)}

Resolve an issue with \texttt{\mathchoice} and localalphabets

The code for keeping a number of math alphabets local (introduced in 2021; see [3]) used \texttt{\aftergroup} to do some cleanup actions after a formula had finished. Unfortunately, \texttt{\aftergroup} can’t be used inside the arguments of the \texttt{\mathchoice} primitive and as a result one got low-level errors if the freezing happened in such a place. The implementation was therefore revised to avoid the \texttt{\aftergroup} approach altogether.

\texttt{(github issue 921)}

Reporting of unused global options when using key/value processing

Using the new key/value option processor did not properly report any unused global options when it was used in handling class options. This has now been corrected.

\texttt{(github issue 938)}

Changes to packages in the graphics category

Fix a \texttt{\mathcolor} bug

The \texttt{\mathcolor} command introduced in [4] needs to scan for following sub- and superscripts, but if it did so at the end of an alignment cell, e.g., in a \texttt{array} environment, the \& was evaluated too early, causing some internal errors. This is now properly guarded for.

\texttt{(github issue 901)}

Changes to packages in the tools category

\texttt{array}: Correctly identify single-line m-cells

Cells in m-columns that contain only a single line are supposed to behave like single-line p-cells and align at the same baseline. To test for the condition, \texttt{array} used to compare the height of the cell to the height of the strut used for the table rows. However, the height of that strut depends on the setting of \texttt{\arraystretch} and if you made this negative (or very large) the test came out wrong. Therefore, we now test against the height of a normal strut to ensure that single-line cells are correctly identified as such (unless their content is truly very tall, in which case aligning is pointless anyway).

\texttt{(github issue 766)}

References

[2] \LaTeX\ Project Team: \TeX\ \LaTeX\ \TeX\ news 32.
  \url{https://latex-project.org/news/latex2e-news/ltnews32.pdf}
[3] \LaTeX\ Project Team: \TeX\ \LaTeX\ \TeX\ news 34.
  \url{https://latex-project.org/news/latex2e-news/ltnews34.pdf}
[4] \LaTeX\ Project Team: \TeX\ \LaTeX\ \TeX\ news 35.
  \url{https://latex-project.org/news/latex2e-news/ltnews35.pdf}
[5] \LaTeX\ Project Team: \TeX\ \LaTeX\ \TeX\ font selection.
  \url{https://latex-project.org/help/documentation/}
New functionality offered as part of the “\LaTeX Tagged PDF” project

We have now enabled new automatic tagging functionality for additional \LaTeX elements, among them most display environments, standard sectioning commands, content, figure and table listings, floats and graphics and bibliographies. This can be activated through

\DocumentMetadata{testphase=phase-III}

At this point in time tagging support is only available for a restricted set of documents, i.e., those that use one of the basic document classes (article, report, and book) and only use commands and environments described in Lamport’s \LaTeX manual.

Using other document classes or adding additional packages in the preamble may work (or may partially work) but at this stage it is not very likely, at least not for packages or classes that excessively alter internals of \LaTeX.

Also note that there are still several environments and commands described in the \LaTeX manual that do not have tagging support yet, notably tabulars, tabbing, the various math environment and a few others. They will get this support as part of phase-III, but some of them will be delayed until after the June release.

A prototype for math tagging (including support for the amsmath environments) is already available, but it is mainly intended for experimentation and feedback and the resulting tagging is by no means the way we envision it to be eventually. If you would like to try it out use the following line:

\DocumentMetadata{testphase=phase-III,math}

Note that the math tagging code at this point in time will clash with packages that redefine the $ character (which then may lead to strange errors) and that packages that use math mode for non-mathematical constructs may result in surprising output as far as tagging is concerned. Feedback on which packages fail with the code in one or another way would be appreciated.

The \texttt{latex-lab} bundle contains various (still un-tagged) documentation files about the new code that can be accessed with \texttt{texdoc -l latex-lab}.
Feedback is welcome! Please use https://github.com/latex3/latex2e/discussions/1010.

New or improved commands
Extending hooks to take arguments
Hooks have always been containers for code whose outcome was entirely dependent on the contents of the hook alone. If any type of contextual information had to be passed to the hook, it had to be done by setting some variable before the hook so that the code in the hook could use that. But this is somewhat hard to keep track of, clumsy to implement, and it required the programmer to have some kind of “hook before the hook” to do that setup.

To make things a bit easier, \texttt{lthooks} was enhanced to support hooks with arguments. Hooks can now be declared and used with arguments, then the code added to these hooks can reference the hook’s arguments using \texttt{#1}, \texttt{#2}, etc., so now hooks can behave more like macros than like \texttt{token lists} (using expl3 terminology). Regular argument-less hooks continue to work exactly like they did before: this extension is completely compatible with older documents and packages.

To declare a hook with arguments, use
\begin{verbatim}
\NewHookWithArguments {<hook>} {<num-args>}
\end{verbatim}

then, similarly, to use the code in the hook, supposing a hook declared with 2 arguments, write
\begin{verbatim}
\UseHookWithArguments {<hook>} {2} {<arg1>} {<arg2>}
\end{verbatim}

Or, if you want to add some code to a hook that takes arguments, write
\begin{verbatim}
\AddToHookWithArguments {<hook>} {[<label>]} {<code>}
\end{verbatim}
everything is the same as for regular hooks, except that the \texttt{<code>} can use the arguments by referencing \texttt{#1}, \texttt{#2}, etc. In this case, if you want to add a parameter to the \texttt{<code>}, you have to double it, as usual.

Additionally, if you want to add “regular” code to a hook with arguments, you can still work \\texttt{AddToHook} — in that case \texttt{#} tokens are not doubled. This means that a package author can decide to add arguments to an existing hook without worrying about compatibility: \texttt{AddToHook} will do the right thing and will not mistakenly reference the newly added arguments.

The commands \texttt{\NewReversedHookWithArguments}, \texttt{\NewMirroredHookPairWithArguments}, \texttt{\AddToHookNextWithArguments}, \texttt{\UseOneTimeHookWithArguments}, and the expl3 counterparts of the commands discussed in this section were also added. The complete documentation can be found in the \texttt{lthooks} documentation [2].

Generic \texttt{cmd} hooks with arguments: Along with the possibility of passing arguments to a regular hook as discussed above, generic \texttt{cmd} hooks can now access the arguments of the command they are patched into, using the interface described in the previous section.

For example, if you were to add some code to the \texttt{\title} command using hooks, you could access the actual title given in the argument. Thus, to write the title of the document in the terminal you could use:
\begin{verbatim}
\AddToHookWithArguments{cmd/title/before} {	exttt{\typeout{Document title: #1}}} \end{verbatim}

As with regular hooks, code added to a \texttt{cmd} hook using \texttt{\AddToHook} will not be able to access the command’s arguments. This means that, as with regular hooks, this change is completely backwards compatible, so previous usages of \texttt{cmd} hooks will work exactly as they did before.

Providing copy and show functions for environments
To copy a command definition we introduced \texttt{\NewCommandCopy} in 2022. This even allows you to copy commands that consist of several internal components, such as robust commands or those with a complex signature. To do the same with environments, e.g., to define the environment \texttt{myitemize} to be equivalent to \texttt{itemize}, you can now write
\begin{verbatim}
\NewEnvironmentCopy{myitemize}{itemize}
\end{verbatim}

There are also \texttt{\Renew} and \texttt{\Declare}, which may be useful depending on the circumstances.

In addition, we offer a \texttt{\ShowEnvironment} command, which displays the \texttt{\begin} and \texttt{\end} code of the environment passed as an argument. E.g., \texttt{\ShowEnvironment{center}} results in the following output:
\begin{verbatim}
» \begin{center}=environment:
» ->\trivlist \centering \item \relax .
» \begin{center} » [
» \begin{center} » ...
» \begin{center} » 
» \end{center}:
» \begin{center} » ->\endtrivlist .
» \begin{center} » \relax .
» \begin{center} » (github issue 963)
\end{verbatim}

\texttt{\IfFileAtLeastTF}
The 2020-10-01 B\TeX{} release introduced the CamelCase tests \texttt{\IfClassAtLeastTF} and \texttt{\IfPackageAtLeastTF} for checking class and package dates. We have now added \texttt{\IfFileAtLeastTF} to allow the same to happen for generic files which contain a \texttt{\ProvidesFile} line. \texttt{\IfFileAtLeastTF}
\begin{verbatim}
\DeclareLowerCaseMapping,
\DeclareTitleCaseMapping and
\DeclareUpperCaseMapping
\end{verbatim}
The move from a case-changing approach using \texttt{\lcname} and \texttt{\ucname} data to one where information is stored by a kernel-managed structure left a gap in the ability of
the user to tune the case changing outcomes. This has now been addressed by the addition of three commands

- \DeclareUppercaseMapping
- \DeclareTitlecaseMapping
- \DeclareLowercaseMapping

which can be used to customise the outcome for codepoints. This can be applied generally or to a specific locale (see also the next section). A small number of pre-defined customisations have been set up in the kernel where the outcomes for pdft\TeX{} should be different for those from Unicode engines. For example

\begin{verbatim}
\DeclareUppercaseMapping{"01F0}{\v{J}}
\end{verbatim}

allows \v{J} to be produced in 8-bit engines: without this allows \v{J} to be produced in 8-bit engines: without this to show up in the index—

\verb/'prog 'my input''

is rendered as `\textit{\texttt{prog 'my input'}}` and not as `prog 'my input'` as preferred by many programmers. This can be adjusted, for example, with the upquote package, which results in the second output. However, for historical reasons doc had its own definition of

---

Extension of the \texttt{\label} command
Previously, in standard \TeX{}, the \texttt{\label} command wrote a new label declaration into the .aux file and stored two values in the second argument of this new label command: \texttt{\currentlabel}, which normally contains the state of the current counter and \texttt{\thepage} for the current page number.

The packages hyperref and nameref then patched the \texttt{\label} command to store five values instead. In addition to the above they saved \texttt{\currentlabelname}, which normally contains the current title text and can be retrieved with \texttt{\nameref}, and \texttt{\currentHref}, which is the name of the destination needed to create an active link. The fifth argument was only used if external references were loaded with the \texttt{xr-hyper} package.

Starting with this release, the number of values stored in \texttt{\newlabel} has been unified. \texttt{\label} now writes a \texttt{\newlabel} command that always contains five values in the second argument (each in a brace group): \texttt{\currentlabel}, \texttt{\thepage}, \texttt{\currentlabelname}, \texttt{\currentHref}, and \texttt{\currentreservedlabeldata} (which is reserved for the kernel).

Additionally, a hook with the name \texttt{\label} has been added. It takes one argument: the label string. Code added to the hook can refer to this argument with \#1. The hook is executed directly before the \texttt{\label} command writes to the .aux file but after the \texttt{\bsphack} command has done its spacing magic, and it is located inside a group; thus, its code only affects the write operation.

---

Code improvements

Performance in checking file existence
The addition of hooks, etc., to file operations had a side effect of making multiple checks that the file existed. In larger documents using many files, these file system operations caused non-trivial performance impact. We now cache the existence of files, such that these repeated filesystem calls are avoided.

\texttt{\verb/'doc Handle \LaTeX{} correctly in the index'}

Due to some problems in the code it wasn’t possible to prevent \LaTeX{} from showing up in the index—\texttt{\DoNotIndex{\LaTeX}}, etc. had no effect. This has now been corrected. (\texttt{\verb/'doc Handle \LaTeX{} correctly in the index'})

\texttt{\verb/'doc Support the upquote package'}

The default quote and backquote characters in typewriter fonts are typographical quotes, e.g., the input

\begin{verbatim}
\verb/'prog 'my input''/
\end{verbatim}

is rendered as `\textit{\texttt{prog 'my input''}}' and not as `prog 'my input'` as preferred by many programmers. This can be adjusted, for example, with the upquote package, which results in the second output. However, for historical reasons doc had its own definition of
\verb and \verbatim and as a consequence the two packages did not cooperate. This has now been fixed and loading upquote together with \texttt{doc} has the desired effect.  
\hfill \textit{(github issue 953)}

\textbf{Default definition for \texttt{do}}

The command \texttt{do} with its nice public name is in reality an internal command inherited from plain \TeX for list processing. However, it only got a definition when \texttt{\begin{document}} was executed, with a result that a user definition in the preamble was unconditionally overwritten at this point. To properly alert the user that this command is not freely available we now make a definition in the format, so that \texttt{\newcommand} and friends produce a proper error message instead of allowing a definition that doesn’t last.  
\hfill \textit{(github issue 975)}

\textbf{New key for filecontents}

The \texttt{filecontents} environment warns on the terminal if a file gets overwritten even if that is intentional, e.g., when you write a temporary file over and over again. To make the warning less noisy in this case we added a new \texttt{nowarn} key that redirects the overwriting warning to the transcript file. We think that some record of the action is still required to help with debugging, thus it is not completely silenced. The warning that nothing gets written, because the file already exists (and the \texttt{force} key was not used), is not altered and still shows up on the terminal.  
\hfill \textit{(github issue 958)}

\textbf{A further hook for shipping out pages}

Since October 2020 the shipout process offers a number of hooks to adjust what is happening before, during, and after the \texttt{\shipout}. For example, with the \texttt{\shipout/before} hook, packages can reset code they have altered (e.g., \texttt{\catcodes} during verbatim-like processing) and with \texttt{\shipout/background} and \texttt{\shipout/foreground} material can be added to the pages. Details are given in \cite{1}.

However, still missing was a hook that allows a package writer to manipulate the completed page (with foreground and background attached) just before the actual shipout happens. For this we now provide the additional hook \texttt{shipout}. One use case (sometimes needed in print production) is to mirror the whole page via \texttt{\reflectbox} including all the extra data that may have been added into the fore- or background.  
\hfill \textit{(github issue 920)}

\textbf{Displaying release information in the \texttt{.log}}

\LaTeX displays its release information at the very beginning of the \LaTeX run on the terminal, and also writes it to the transcript file if that is already opened at this point. While this is normally true, it is not the case if the \LaTeX run was started passing additional \TeX code on the command line, e.g.,

\verb pdflatex '{\PassOptionsToClass{11pt}{article} \input{myarticle}'}

In this case the release information is displayed when \texttt{\PassOptionsToClass} is processed but the transcript file is only opened when the output file name is known, i.e., after \texttt{\input} has been seen, and as a result the release information is only shown on the terminal.

To account for this scenario, we now repeat the release information also at the very end of the transcript file where we can be sure that it is open and ready to receive material.  
\hfill \textit{(github issue 944)}

\textbf{Bug fixes}

\textbf{Incompatibility between \texttt{doc} and unicode-math}

The unicode-math package alters the catcode of \texttt{|} but does not adjust its value for use in \texttt{doc}, with the result that “or” modules, i.e., \texttt{(A|B)} are displayed in a strange way. This is now fixed with some.firstname code that will eventually be moved into \texttt{unicode-math}.  
\hfill \textit{(github issue 820)}

\textbf{A fix for \texttt{\hspace}}

The change to \texttt{\hspace}, done in 2020 to make it \texttt{calc}-aware, had the unfortunate side effect that starting a paragraph with \texttt{\hspace} would result in the execution of \texttt{\everypar} inside a group (i.e., any local changes would immediately be revoked, breaking, for example, \texttt{wrapping} in that special situation). This got fixed with the 2022-11 PL1 hotfix, so was already corrected in the previous release, but is only now documented in the newsletter.  
\hfill \textit{(github issue 967)}

\textbf{Ensure that \texttt{\cs} is defined in \texttt{ltxdoc}}

The class \texttt{ltxdoc} defined the command \texttt{\cs} to typeset a command name with a backslash in front. This definition was moved to the \texttt{doc} package itself. This meant that it was suddenly missing when reverting to the old \texttt{doc} package implementation via the class option \texttt{doc2}. This has now been corrected.  
\hfill \textit{(github issue 981)}

\textbf{Improve spacing at top of minipages}

A list and several other document elements add some vertical space in front of them. However this should not happen at the beginning of a box (such as a \texttt{minipage}) and normally it doesn’t, because \TeX automatically drops such space at the start of a vertical list. However, if there is some invisible material, such as a \texttt{\color} command, a \texttt{hyperref} anchor, a \texttt{\write} or other such items, then the list is no longer empty and \TeX no longer drops the vertical space.

With the new paragraph handling introduced in 2021 it is now finally possible to detect and avoid this problem and apply appropriate counter measures so that from now on the spacing will always be correct.  
\hfill \textit{(github issue 989)}
A fix for \NewCommandCopy and \ShowCommand
When copying and showing definitions of (non-expandable) document commands (a.k.a. commands defined by \NewDocumentCommand and friends) containing empty or only m-type arguments, these commands were wrongly recognized as expandable ones. This is fixed in the present \LaTeX{} release.  \(\text{(github issue 1009)}\)

Corrections for switching math version
Some internal code improvements improve support for switching math version when nested within an outer math expression. This will improve \boldsymbol{} and similar commands.  \(\text{(github issue 1028)}\)

Allow par as a filename
\input{par} or \includegraphics{par} could give spurious errors. This has been fixed by making an internal command \long. \(\text{(github issue 942)}\)

Correct setting of \endlinechar in ++ arguments
In the particular case of a document command with a ++-type argument used inside \ExplSyntaxOn/Off, newlines would be misinterpreted as spaces because the value of \endlinechar was set too late. This has been fixed, and now newlines are correctly translated to "the character ^^M". \(\text{(github issue 876)}\)

Correct handling of hooks with only 'next' code
When \AddToHookNext was used on a not-yet-declared hook, that hook would be incorrectly identified as empty by \ShowHook. Also, if that hook was later declared, that ‘next’ code would not be executed. This has been fixed by correctly initializing the hook structure when \AddToHookNext is used. \(\text{(github issue 1052)}\)

Ignoring space after $$
Space is normally ignored after a closing $$, but internal \LaTeX{} font handling code could interfere if \leqno was used. \leqno and \leqno have been redefined to add \ignorespaces after the math group. \(\text{(github issue 982)}\)

Documentation improvements
Updates to the guides
When \LaTeX{} 2e was released, the team provided documentation for both document authors and package/class developers in the two files \usrguide{} and \clsguide{}. Over time, the team have augmented these documents as new methods have been added to the kernel. However, they retained their structure as assuming familiarity with \LaTeX{} 2.09. This meant that for new users, there was material which is no longer relevant, and less clarity than desirable regarding the approaches that are recommended today.

The two files have now been (partially) re-written, with the versions available previously now frozen as \usrguide-historic{} and \clsguide-historic. More material has been carried forward in the class/package guide than in the user guide, but both are worth a re-read by experienced \LaTeX{} users.

Displaying the exact release dates for \LaTeX{}
In some situations it is necessary to find out the exact release dates for older versions of the \LaTeX{} format, for example, when you need to use different code in a package depending on the availability of a certain feature and you therefore want to use \IfFormatAtLeastTF{(date)} or the rather horrible construction \@if@ft@r\fmtversion{(date)}, if you want to cater for formats that are older than 2020.

Or you know that your package is definitely not going to work with a format before a certain \(\langle date\rangle\), in which case you could use \NeedsTeXFormat{LaTeX2e}{\langle date\rangle}\ to ensure that users are alerted if their format is too old.

The big problem is knowing the exact \(\langle date\rangle\) to put into such commands; in the past, that was not that easy to find. You could have looked in the file \changes.txt, but that is hidden somewhere in your installation and if you try \texttt{texdoc \textasciitilde changes.txt} you get more than thirty results and the right file is by no means the first.

Yukai Chou (@muzimuzhi) kindly provided a patch for this, so that we now have the exact dates for each \LaTeX{} format listed in an easy to remember place: in \ltnews.pdf and that file conveniently also contains all major features and changes to \LaTeX{} over the years—one of which is most likely the reason you need the \(\langle date\rangle\) in the first place.

The date is now given in parentheses in the newsletter title, thus this newsletter tells you that on 2023-06-01 the command \NewEnvironmentCopy, a new \shipout hook, etc. was made available. And looking into \ltnews.pdf you can now easily find out that the \LaTeX{}X programming layer was added on 2020-02-02 (because the date was so nice) and not on the first of the month. \(\text{(github issue 982)}\)

Fresh from the press: “The \LaTeX{} Companion, third edition” is now in print
The third edition of \emph{The \LaTeX{} Companion}, third edition is now available. This is the result of five years of careful work and we hope that it will provide our readers with all the information they need to successfully navigate the \LaTeX{} ecosystem and efficiently produce beautiful documents.

Since the publication of the last edition (2004), a lot has happened in the \LaTeX{} world and thus a complete rewrite was necessary. All chapters have been thoroughly revised, and in many cases significantly extended, to describe new important functionality and features. More than 5,000 add-on packages have been analyzed in detail, out of which roughly 10\% have been chosen for inclusion in \emph{The \LaTeX{} Companion}. All important aspects of these packages are described to provide the user once
again with a satisfying one-stop-shop experience for the
decade to come.
To cover what we thought worth describing today,
the book nearly doubled in size. The print edition is
therefore produced as a two-volume set and sold as a
bundle. Both volumes come as hardcover with ribbons
to easily mark pages in the book.
To give you an idea of what is covered in the third
edition you can find some excerpts at

https://www.latex-project.org/news/
2023/03/17/TLC3

The edition is also available as an eBook (Parts I and II
combined) consisting of PDF and ePub format, without
DRM. Finally, the publisher offers the combination of
the printed books and the digital versions at a very
attractive price not available anywhere else.

Changes to packages in the tools category
multicol: Better support for CJK languages
The default minimum depth of each column in a
multicols corresponds to the depth of a “p” in the
current font. This helps to get some uniformity if rules
are used between the columns and makes sense for Latin-
based languages. Until now it was hard-wired, but for
CJK (Chinese/Japanese/Korean) languages it is better
to use a zero depth, because there all characters have
the same height and depth. And even with Latin-based
languages one might want to use the depth of a \strut
or that of a parenthesis. So we now offer a way to adjust
this while maintaining backward compatibility: redefine
\multicolmindepthstring to hold whatever you want
to get measured for its depth (the width is not relevant).
(\texttt{github issue 698})

multicol: Fix handling of nested environments
If \texttt{multicols} environments have been nested into
each other (the inner one boxed) it could fail if the
boxed environment appeared near a page break. The
problem was that the output routine was called while
the \texttt{hsize} was still altered to fit the column width
of the inner \texttt{multicols} — thereby messing up the
placement of columns of the page. This has now been
fixed. (\texttt{github issue 1002})

References
[1] Frank Mittelbach, \LaTeX Project Team:
The \texttt{ltshipout} documentation.
Run \texttt{texdoc ltshipout-doc} to view.

[2] Frank Mittelbach, Phelype Oleinik,
\LaTeX Project Team: \LaTeX's hook management.
Run \texttt{texdoc lthooks-doc} to view.
Contents

News from the “\LaTeX{} Tagged PDF” project 93
   Approaching an important milestone . . . . . . 93
   A GitHub repository dedicated to the project . . 93

Hooks, sockets and plugs 93

Document properties and cross-referencing 94

New or improved commands 95
   Testing for the \LaTeX{}3 programming layer
      version: \texttt{\IfExplAtLeastTF} . . . . . . . . 95

Code improvements 95
   Support for tabs in \verb* and \verbatim* . . . 95
   Improved argument checking for box commands 95
   Aligning status of tilde with other active
      characters . . . . . . . . . . . . . . . . . . . 95
   In the programming layer . . . . . . . . . . . . 96

Removed kernel commands 96

Changes to packages in the tools category 96
   \texttt{longtable}: correct p-column definition . . . 96

News from the “\LaTeX{} Tagged PDF” project

The multi-year project to automatically tag \LaTeX{} documents in order to make them accessible [3] is progressing steadily (at this point in time mainly as experimental \texttt{latex-lab} code).

Just recently we added support for automatic tagging of tabular structures including environments from \texttt{tabularx} and \texttt{longtable}. The code is still in its early stages and lacks configuration possibilities—these will be added in the future.

Approaching an important milestone

Nevertheless, with this new addition we are more or less able to automatically tag any document that confines itself to the commands and environments described in Leslie Lamport’s \texttt{\LaTeX{} Manual} [1] by simply adding a single configuration line at the top.

In addition, a number of extension packages that go beyond Lamport are already supported, most importantly perhaps \texttt{amsmath} (providing extended math capabilities) and \texttt{hyperref} (enhancing \LaTeX{} with interactive hyperlinking features). Also already supported are some of the major bibliography support packages such as \texttt{natbib} and \texttt{biblatex}.

For now activation is done through the line

\begin{verbatim}
\DocumentMetadata
   \{testphase={phase-III,math,table}}
\end{verbatim}

The math and the tabular support are not yet incorporated into phase-III but need their own activation, so that we can better experiment with additions and code adjustments.

The \texttt{latex-lab} bundle contains various (still un-tagged) documentation files about the new code that can be accessed with \texttt{texdoc -1 latex-lab}.

A GitHub repository dedicated to the project

We have also started a new GitHub repository mainly intended for reporting issues, and offering a platform for discussions. For example, there is one discussion on ways to extend the \LaTeX{} \texttt{tabular} syntax to allow describing the logical structure of tables (e.g., which cells are header cells, etc.).

Having all issues and discussions related to the project in a single place instead of being spread across multiple repositories such as \texttt{latex2e}, \texttt{latex3}, \texttt{tagpdf}, \texttt{hyperref}, \texttt{pdfresources}, etc., helps people to find information easily and report any issue related to the project without needing to know in which code repository the problematic code resides.

You find this repository at \url{https://github.com/latex3/tagging-project} and the mentioned discussion on \texttt{tabular} syntax at \url{https://github.com/latex3/tagging-project/discussions/1}.

Your feedback is important and reporting what doesn’t yet work is beneficial to all users, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

Hooks, sockets and plugs

In previous releases of \LaTeX{} we introduced the general concept of hooks (both specific and generic ones). These are places in the code into which different packages (or the user in the document preamble) can safely add their own code to extend the functionality of existing commands and environments without the need to overwrite or patch them in incompatible ways. An important feature of such hooks is that the code chunks added by different packages can be ordered by rules, if necessary, thereby avoiding problems arising from
differences in package loading order. See \textit{\LaTeX} News issues 32–34 \cite{texdoc-sockets-doc} for more information.

However, sometimes you need a kind of “hook” into which only a single chunk of code is placed at any time. For example, code that implements footnote placement in relation to bottom floats (above or below them). But at any time in the document only one such placement code can be in force. Or consider the extra code needed for making \textit{\LaTeX} documents accessible (e.g., adding tags to the PDF output). Such code is either there (perhaps in alternative versions) or not at all, but it cannot have code from other packages added at the same point interfering with the algorithm.

For these use cases we now introduce the concept of sockets and plugs. A socket is a place in the code into which one can put a plug (a chunk of code with a name) after which the socket is in use; to put in a different plug, the former one has to be taken out first\footnote{While this is in theory possible to model with the existing hook mechanism, it is inefficient and cumbersome.}. A socket may or may not have inputs that can then be used by the plugs. While this is technically not much different to putting a command in the code and at some point alter its definition, the advantage is that this offers a consistent interface, allows for status information, supports tracing, etc.

You declare a new socket and possibly some plugs for it with

\begin{verbatim}
\NewSocket{⟨socket name⟩}{⟨# of inputs⟩}
\NewSocketPlug{⟨socket name⟩}{⟨plug name⟩}{⟨code⟩}
\end{verbatim}

For example, after the declaration \texttt{\NewSocket{foo}{0}} you can immediately use this socket in your code with \texttt{\UseSocket{foo}}. The \texttt{\NewSocket} declaration automatically defines a simple plug with the name \texttt{noop} for the socket and assigns it to the socket (plugs it in), thus your \texttt{\UseSocket} sits idle doing nothing\footnote{Think of electric outlets and plugging something into them.} until you assign it a different plug, which is done with \texttt{\AssignSocketPlug}. This takes the current plug out and puts the new one in. All the declarations and commands are also available in the \textit{\LaTeX}3 programming layer as \texttt{\socketnew:n, \socketnewplug:n}, etc.

With this concept we can, for example, add tagging support for the “\textit{\LaTeX} Tagged PDF” project to various packages without altering their behavior if the tagging code is inactive. Activating one or the other form of tagging then just means to assign named plugs to the different sockets.

This is just a brief introduction to the mechanism; for more detailed documentation see \texttt{texdoc\ ltsockets-doc}.

\section*{Document properties and cross-referencing}

Traditional \textit{\LaTeX} uses \texttt{\label{(key)}} to record the values of two “local” properties of the document: the textual representations of the current page number and the current \texttt{\ref} value set by \texttt{\refstepcounter} declarations \cite[p. 209]{texdoc-sockets-doc}. (These declarations are issued, for example, by sectioning commands, by numbered environments like \texttt{equation}, and by \texttt{\item} in an \texttt{enumerate} or similar environment.)

These two recorded values can then be accessed and typeset (from anywhere in the next run of the document) by use of the (non-expandable) commands \texttt{\ref} and \texttt{\pageref} using the \texttt{key} that was specified as the argument to \texttt{\label} when recording these values. This supported basic cross-referencing (within a document), using these recorded values to provide both page-related and counter-related information (such as the page xvii or the subsection number 4.5.2)\footnote{In the Spring 2023 release of \textit{\LaTeX}, the \texttt{\label} command was extended to record, in addition, both a title (such as the text used in a section head) and the \texttt{logical name} used for an associated link target provided these have been set by packages such as \texttt{nameref} or \texttt{hyperref}.}.

Over the years \textit{\LaTeX} packages have appeared that extend this basic “label-ref system” in various ways. For example, the \texttt{refcount} package made a small but significant change to the functions used to access recorded values, by making them expandable. And the \texttt{smart-ref} package supports the storage of a larger collection of counter values so that, for example, a cross-reference can refer to the relevant chapter together with an equation tag. The \texttt{cleveref} package stores (by means of a second, internal “logical label”) extra information such as the name of the counter. The \texttt{hyperref} package adds the \texttt{\autoref} command, which tries to retrieve the name of a counter from the \texttt{logical name} used for a link target. The \texttt{tikzmarks} library records information about \texttt{labelled positions} on the page when using \texttt{tikz}.

Finally, the \texttt{zref} package implements many related ideas, including a general idea of properties and lists of properties, with methods to record, and subsequently access, the value of any declared property.

Starting with this release, the \textit{\LaTeX} kernel provides handling of general document properties as a core functionality with standard interfaces. This is based on concepts introduced by the \texttt{zref} package but with some differences in detail, particularly in the implementation. It supports the declaration of new properties, and the recording of the values of any list of properties. These values are retrieved expandably.

To set up a new property that is the current chapter number, for example, here is the declaration to use.

\begin{verbatim}
\NewProperty{chapter}{now}{?}{\thechapter}
\end{verbatim}
The second argument means that the property value will be recorded immediately (“now”), and not “during the next \shipout”. The third argument sets a default to be used when, for example, an unknown label is supplied. The final argument contains the code that will, as part of the recording process, be expanded to obtain the value to record for this property.

Then, to record the value of this new property, together with others, use this command.

\RecordProperties{mylabel}{chapter,page,label}

This records the current values for the properties chapter, page, and label, using mylabel as the label, or key, for the record.

To reference (i.e., retrieve) this recorded value for use in a cross-reference to this chapter, use the \RefProperty{mylabel}{chapter} command with two arguments: the label, or key, and the property.

\RefProperty{mylabel}{chapter}

The \LaTeXX kernel itself contains declarations for some generally useful properties, including these:

- **label**: the textual representation of the current \ref value, see above;
- **page**: the textual representation of the page number for the page currently under construction;
- **title**: the title, if set by, e.g., \nameref;
- **target**: the logical name of the associated link target, if set by, e.g., \hyperref;
- **pagetarget**: the logical name of the target added by \hyperref at the origin of each shipped out page;
- **pagenum**: the value of the \LaTeX counter page in Arabic numerals;
- **abspage**: the absolute page number of the page under construction, i.e., one more than the number of pages shipped out so far (thus it starts at 1 and is increased by 1 whenever a page is shipped out);
- **counter**: the name of the counter that produced the current \ref value, i.e., the counter that was stepped in the most recent \refstepcounter within the current scope;
- **xpos**, **ypos**: the position on the shipped out page as set by the most recent \pdfsavepos: recording these properties should be done as soon as possible after saving the position.

Both \LaTeXX 2\varepsilon commands (using camel-case names) and \LaTeXX3 programming layer commands are provided. For a more complete documentation, see texdoc ltproperties-doc.

### New or improved commands

#### Testing for the \LaTeXX3 programming layer version: \IfExplAtLeastTF

The integration of expl3 (the \LaTeXX3 programming layer) into the kernel means that programmers can use all of the features available without needing to load it explicitly. However, as expl3 is upgraded separately from \LaTeXX 2\varepsilon and is not a separate package, its version is different from that of \LaTeXX 2\varepsilon and cannot be tested using \IfPackageAtLeastTF. To date, low-level methods have therefore been needed to check for the availability of features in expl3. We have now added \IfExplAtLeastTF as a test working in the same way as \IfPackageAtLeastTF but focused on the pre-loaded programming layer. Programmers can check the date of expl3 they are using in the .log, as it appears both at the start and end in the format

L3 programming layer <YYYY-MM-DD>

just after the line which identifies the format (LaTeX2e, etc.).

(\github\ issue 1004)

#### Code improvements

**Support for tabs in \verb* and \verbatim**

\LaTeXX converts a single tab to a single space, which is then treated like a “real” space in typesetting. The same has been true to date inside \verb, but was done in a way that meant that they remained as normal spaces even in \verb*, etc. We have now adjusted the code so that tabs are retained within the argument to \verb and \verb*, and the \verbatim and \verbatim* environments, independently from spaces, and are set up to print in the same way spaces do. This means that they now generate visible spaces inside \verb* and \verbatim*, and their behavior can be adjusted if required to be different from that of spaces.

(\github\ issue 1085)

**Improved argument checking for box commands**

Previously if an alignment option had an unexpected value, such as \makebox[4cm][x]{text}, no warning was given but the box content was silently discarded. This will now produce a warning and act like the default c alignment. \framebox and \parbox have a similar change.

(\github\ issue 1072)

**Aligning status of tilde with other active characters**

Some time ago we revised the definition of active characters in pdf\TeX to allow the full range of UTF-8 codepoints to be used in for example labels, file names, etc. However, ~ was not changed at that point as it is active independent of the engine in use. This has now been corrected: the definition of ~ is an engine-protected one which gives the string version of the character if used inside a cname.
In the programming layer

In the programming layer (expl3), we have revised the behavior of the titlecasing function to enable this to either titlecase only the first word of the input, or to titlecase every word. This should be transparent at the document level but will be useful for programmers.

We have also added the ability to define variables and functions inside \fpeval (at the expl3 level this is \fp_eval:n). This allows programmers to create non-standard functions that can then be used inside \fpeval. For example, this could be used to create a new function dinner:

\ExplSyntaxOn
\fp_new_variable:n{duck}
\fp_new_function:n{dinner}
\fp_set_function:nnn{dinner}{duck}
\{duck - 0.25 * duck\}
\fp_set_variable:nn{duck}{1}
$\fp_eval:n{duck}$
\fp_eval:n{dinner(duck)}
\fp_set_variable:nn{duck}{dinner(duck)}
\fp_eval:n{dinner(duck)}
\fp_set_variable:nn{duck}{dinner(duck)}
\fp_eval:n{dinner(duck)}
\fp_set_variable:nn{duck}{dinner(duck)}
$\fp_eval:n{dinner(duck)}$
\ExplSyntaxOff

The computation above would then generate

\[
1 > 0.75 > 0.5625 > 0.421875 > 0.31640625
\]

Users will be able to access added functions without needing to use the expl3 layer. It is possible that a future release of \LaTeX will add the ability to create and set floating point variables at the document level: this will be examined based on feedback on the utility of the programming layer change.

Changes to packages in the tools category

longtable: correct p-column definition

In general the longtable implementation follows the array usage but the package didn’t take over a change made 1992 in array which adjusted the handling of the strut inserted at the begin of p-columns. As a consequence there are a number of inconsistencies in the output of p-columns between tabular and longtable. This has been corrected; longtable now uses for the strut the same definition as array.  

(\textit{github issue 1128})

References


[2] \LaTeX Project Team. \LaTeX2ε news 1–38.  

https://latex-project.org/publications/indexbyyear/2020/

Removed kernel commands

It is very rare that commands are removed from the \LaTeX kernel. However, in this release we have elected to remove \GetDocumentCommandArgSpec, \GetDocumentEnvironmentArgSpec, \ShowDocumentCommandArgSpec and \ShowDocumentEnvironmentArgSpec from the kernel. These commands have been moved back to the “stub” xparse provided in l3packages. The reason for this change is that the removed commands exposed implementation details. They were essentially debugging tools which with hindsight should not have been made available directly in the kernel.
Contents

Introduction .......................... 97

News from the “\LaTeX\ Tagged PDF” project .......................... 97

Enhancements to the new mark mechanism ......................... 98

Providing \xtemplate in the format ........................................ 98

New or improved commands ................................................. 99
  \texttt{doc}: Provide \texttt{\ProvideDocElement} ......................... 99
  \texttt{doc}: Better support for \texttt{upquote} .......................... 99
  \texttt{ifthen}: Allow active characters in comparisons ............. 99
  New conditionals: \texttt{\IfClassAtLeastT} and friends .............. 99

Code improvements ....................................................... 99
  Load packages only at the top level ................................ 99
  Keep track of lost glyphs ............................................. 100
  Improve \texttt{fontenc} error message ................................ 100
  Warn if counter names are problematic .............................. 100
  Extended information in \texttt{\listfiles} ............................ 100
  Optimize creation of simple document commands .................. 100
  Handling of end-of-lines in +v arguments of \texttt{\NewDocumentCommand} and friends .............................. 100
  Declaring appropriate sub-encodings for TS1 symbol fonts ... 100
  Behavior when loading \texttt{textcomp} without options .......... 101
  Rollback improvements ............................................... 101

Documentation improvements ............................................ 101
  Further updates to the guides ....................................... 101

Bug fixes ............................................................... 101
  Fix inconsistent expansion of the package option list .......... 101
  Fix logic for first mark (page region) ............................ 102
  Struts at the end of footnotes or p columns ....................... 102
  Fix a “missing \texttt{item}” rollback error ......................... 102

Changes to packages in the \texttt{amsmath} category .............. 102
  \texttt{amsmath}: Correct equation tag placement ................ 102

Changes to packages in the tools category ......................... 102
  \texttt{array}, \texttt{longtable}, \texttt{tabular}: Support tagging ........ 102
  \texttt{array}: No \texttt{\unskip} in math cells .......................... 102
  \texttt{verbatim}: \texttt{\verb} showed visible spaces ............... 102
  \texttt{verbatim}: Support tabs in \texttt{\verbatiminput}\* .......... 102
  \texttt{multicol}: \texttt{\columnbreak} interferes with mark mechanism ....................................................... 102
  \texttt{showkeys}: Allow \texttt{\newline} in amsthm to work ........ 102
  \texttt{xr}: Support links and properties ........................................ 102

Changes to files in the \texttt{cyrillic} category ................... 103
  Correct definition of \texttt{\&k} ..................................... 103

Introduction

The \LaTeX\ Project team remains strongly focused on producing automatically tagged PDF output for accessibility and reuse. At the beginning of 2024 the ISO PDF/UA-2 and the WTPDF (well-tagged PDF) standards were released and we are glad to be able to report that it is now possible to use \LaTeX\ to automatically produce documents that conform to these new standards.\footnote{At the present time we are still in a trial/prototype phase in which only a limited set of document classes and packages are supported. Over the next releases we expect to gradually lift these restrictions and eventually provide the full functionality as part of the core distribution, rather than through \texttt{latex-lab} modules.} A sample collection of such documents ranging from classical texts, such as the Bible, to recent technical papers submitted to arXiv.org can be found at \url{https://github.com/latex3/tagging-project/discussions/72}.

In February Ulrike and Frank presented the current project status during the 5th International Workshop on “Digitization and E-Inclusion in Mathematics and Science 2024” (DEIMS 2024) at Nihon University, Tokyo, Japan; see \cite{8}.

News from the “\LaTeX\ Tagged PDF” project

In the previous \LaTeX\ News \cite{7} we announced some prototype support for tagged tabulars. Some of the necessary code has now been moved from \texttt{latex-lab} to the corresponding packages (using sockets and plugs) and to the \LaTeX\ kernel (for those parts that are also necessary for other aspects of tagging).

The kernel code specific to tagging is implemented in the file \texttt{\tttagging.dtx}. For now it contains \texttt{\UseTaggingSocket}, a special invocation command for...
sockets that are specific to tagging. This enables us to also provide `\SuspendTagging` and `\ResumeTagging`, i.e., a very efficient way to temporarily disable the whole tagging process. This is, for example, necessary if some code is doing trial typesetting. In that case the trials should not generate tagging structures—only the finally-chosen version should. Thus, `tabularx`, for example, stops the tagging while doing its trials to figure out the correct column widths to use, and then re-enables tagging when the table is finally typeset.

Over time, `lttagging.dtx` will hold more general tagging code as appropriate. For now it is only documented as part of `source2e.pdf` but long term we will provide a separate guide for tagging, which will then also include the information currently found in various other places, e.g., `tagpdf.pdf`.

We also added support for a few missing commands described in Leslie Lamport’s `BtTX Manual` [1]: If `phase-III` is used the `\marginpar` command will be properly tagged (depending on the PDF version) as an `Aside` or a `Note` structure. In the standard classes `\maketitle` will be tagged if the additional testphase module `title` is used.

The `math` module has been extended and now includes options to attach MathML files to the structures. First tests with a PDF reader and screen reader that support associated files look very promising. Examples of PDF files tagged with the new method can be found at https://github.com/latex3/tagging-project/discussions/72.

At last various small bugs and problems reported at https://github.com/latex3/tagging-project have been fixed. Such feedback is very valuable, so we hope to see you there and thank you for any contribution, whether it is an issue or a post on a discussion thread.

**Enhancements to the new mark mechanism**

In June 2022 we introduced a new mark mechanism [2, p. 76] that allows keeping track of multiple independent marks. It also properly supports top marks, something that wasn’t reliably possible with `BtTX` before.

There was, however, one limitation: to retrieve the marks from the page data it was necessary to `\vsplit` that data artificially so that `T` would produce split marks that the mechanism could then use. Unfortunately, `T` gets very upset if it finds infinite negative glue (e.g., from `\vss`) within this data. This is not totally surprising because such glue would allow splitting off any amount of material as such glue would hide its size. `T` therefore responds with an error message if it find such glue while doing a `\vsplit` operation (and it does so even if a later glue item cancels the infinite glue).

To account for this, the code in 2022 attempted to detect this situation beforehand and if so did not do any splitting but, of course, it would then also not extract any mark information.

In this release the approach has been changed and we always do a `\vsplit` operation and thus always get the right mark data extracted. While it is not possible to avoid upsetting `T` in case we have infinite negative glue present, it is possible to hide this (more or less) from the user. With the new code `T` will neither stop nor show anything on the terminal. What we can’t do, though, is avoid an error being written to the log file, but to make it clear that this error is harmless and should be ignored we have arranged the code so that the error message, if it is issued, takes the following format:

```
! Infinite glue shrinkage found in box being split. ﹋<argument>∞ ﹋<argument>! Infinite shrink error above ignored ! 1. ... }
```

Not perfect (especially the somewhat unmotivated `<argument>`), but you can only do so much when error messages and their texts are hard-wired in the engine.

So why all this? There are two reasons: we do not lose marks in edge cases any more, and perhaps more importantly we are now also reliably able to extract marks from arbitrarily boxed data, something that wasn’t possible at all before. This is necessary, for example, to support extended marks in `multicols` environments or extract them from floats, marginpars, etc.

Details about the implementation can be found in `texdoc ltmarks-code` or in the shorter `texdoc ltmarks-doc` (which only describes the general concepts and the command interfaces).

**Providing xtemplate in the format**

In `BtTX` News 32, we described the move of one long-term experimental idea into the kernel: the package `xparse`, which was integrated as `ltcmd`. With this edition, we move another long-term development idea to stable status: `templates`.

In this context, templates are a mechanism to abstract out various elements of a document (such as “sectioning”) in such a way that different implementations can be interchanged, and design decisions can be implemented efficiently and controllably.

In contrast to `ltcmd`, which provides a mechanism that many document authors will exploit routinely, templates are a more specialised tool. We anticipate that they will be used by a small number of programmers, providing...
generic ideas that will then be used within document classes. Most document authors will therefore likely directly encounter templates only rarely. We anticipate though that they will be using templates provided by the team or others.

The template system requires three separate ideas

- **Template type**: the “thing” we are using templates for, such as “sectioning” or “enumerated-list”
- A template: a combination of code and keys that can be used to implement a type. Here for example we might have “standard-\texttt{L}\TeX-sectioning” as a template for “sectioning”
- One or more instances: a specific use case of a template where (some) keys are set to known values. We might for example see “\texttt{L}\TeX-section”, “\texttt{L}\TeX-subsection”, etc.

As part of the move from the experimental \texttt{xtemplate} to kernel integration, the team have revisited the commands provided. The stable set now comprises

- \texttt{\newtemplate}
- \texttt{\DeclareTemplateInterface}
- \texttt{\DeclareTemplateCode}
- \texttt{\DeclareTemplateCopy}
- \texttt{\UseTemplateDefault}
- \texttt{\UseTemplate}
- \texttt{\DeclareInstance}
- \texttt{\DeclareInstanceCopy}
- \texttt{\EditInstance}
- \texttt{\UseInstance}
- \texttt{\IfInstanceExistsTF} and variants

To support existing package authors, we have released an updated version of \texttt{xtemplate} which will work smoothly with the new kernel-level code. The existing commands provided in \texttt{xtemplate} will continue to work, but we encourage programmers to move to the set above.

**New or improved commands**

**doc**: Better support for upquote

In \texttt{L}\TeX News 37 [6] we wrote that support for the upquote package was added to the doc package, but back then this was added only for \texttt{\verb} and the \texttt{verbatim} environments. However, in a typical .\texttt{dtx} file, most of the code will be in the body of some \texttt{macroc ode} or \texttt{macroc ode*} environments, and neither of these was affected by adding upquote. We have now updated \texttt{doc} so that upquote alters the quote characters in these environments as well.  

```markdown
(github issue 1230)
```

**ifthen**: Guard against active characters in comparisons

The \texttt{\ifthenelse} command now ensures that <, = and > are safe in numeric tests, even if they have been made active (typically by \texttt{babel} language shorthands).

```markdown
(github issue 756)
```

**New conditionals**: \texttt{\IfClassAtLeastT} and friends

Around 2020 we added a number of conditionals with CamelCase names, i.e., \texttt{\IfClassAtLeastTF}, \texttt{\IfClassLoadedTF}, \texttt{\IfClassLoadedWithOptionsTF}, \texttt{\IfPackageLoadedWithOptionsTF}, \texttt{\IfPackageAtLeastTF}, \texttt{\IfPackageLoadedTF}, and \texttt{\IfPackageLoadedWithOptionsTF} to help arranging conditional code that depends on the release of a particular class, package or format. However, we only provided the TF commands and not also the T and F variants. This has now been changed.

In 2023 we introduced \texttt{\IfFileAtLeastTF} but we did not also provide \texttt{\IfFileLoadedTF} at the same time. This conditional and its T and F variants have now also been added. Remember that one can only test for files that contain a \texttt{\ProvidesFile} line. We did the same for the conditionals \texttt{\IfLabelExistsTF} and \texttt{\IfPropertyExistsTF}, also introduced in 2023.\footnote{By mistake they were initially introduced under the names \texttt{\IfLabelExistTF} and \texttt{\IfPropertyExistTF}; we corrected that at the same time. This is a breaking change, but the commands have been used so far only in kernel code.}

```markdown
(github issues 1222 1262)
```

**Code improvements**

**Load packages only at the top level**

Classes and packages must be loaded only by using the commands \texttt{\documentclass} and \texttt{\usepackage} or the class interface commands such as \texttt{\LoadClass} or \texttt{\RequirePackageWithOptions}; moreover, all of these must always be used at the top level, and not inside a group of any type (for example, within an environment).

Previously \texttt{L}\TeX did not check this, which would often lead to low level errors later on if package declarations were reverted when a group ended. \texttt{L}\TeX now checks the group level and an error is thrown if the class or package is loaded in a group.

```markdown
(github issue 1185)
```
Keep track of lost glyphs
A while ago we changed the \text{T}e\text{X} default value for \texttt{\tracinglostchars} from 1 to 2 so that missing glyphs generate at least a warning, but we forgot to make the same change to \texttt{\tracingnone}. Thus, when issuing that command \text{T}e\text{X} stopped generating warnings about missing glyphs. This has now been corrected. (\texttt{github issue 549})

Improve fontenc error message
If the fontenc package is asked to load a font encoding for which it doesn’t find a suitable \texttt{.def} file then it generates an error message indicating that the encoding name might be misspelled. That is, of course, one of the possible causes, but another one is that the installation is missing a necessary support package, e.g., that no support for Cyrillic fonts has been installed. The error message text has therefore been extended to explain the issue more generally. (\texttt{github issue 1102})

Warn if counter names are problematic
In the past it was possible to declare, for example, \texttt{\newcounter{index}} with the side-effect that this defines \texttt{\theindex}, even though \TeX{} has a \texttt{\theindex} environment that then got clobbered by the declaration. This has now been changed: if \texttt{\the(counter)} is already defined it is not altered, but instead a warning message is displayed. (\texttt{github issue 823})

Extended information in \texttt{\listfiles}
The \texttt{\listfiles} command provides useful information when finding issues related to variation in package versions. However, this has to date relied on the information in the \texttt{\ProvidesPackage} line, or similar: that can be misleading if for example a file has been edited locally. We have now extended \texttt{\listfiles} to take an optional argument which can include the MD5 hash and size of each file in the \texttt{.log}. Thus for example you can use

\texttt{\listfiles[hashes.sizes]}
to get both the file sizes and file hashes in the \texttt{.log} as well as the standard release information. (\texttt{github issue 945})

Optimize creation of simple document commands
Creating document commands using declarations such as \texttt{\NewDocumentCommand}, etc., provides a very flexible way of grabbing arguments. When the document command only takes simple mandatory arguments, this has to-date added an overhead that could be avoided. We have now refined the internal code path such that “simple” document commands avoid almost any overhead at point-of-use, making the results essentially as efficient as using \texttt{\newcommand} for low-level \text{T}e\text{X} constructs. Note that as \texttt{\NewDocumentCommand} makes engine-robust commands, the direct equivalent to \texttt{\newcommand} is \texttt{\NewExpandableDocumentCommand}. (\texttt{github issue 1189})

Handling of end-of-lines in +v arguments of \texttt{\NewDocumentCommand} and friends
The +v argument type provided by declarations such as \texttt{\NewDocumentCommand}, etc., allows grabbing of multiple lines of text in a verbatim-like argument. Almost always, the result of this grabbing will be used in a typesetting context. Previously, the end-of-line characters were stored literally as category code 12 (“other”) ^{\texttt{\texttt{\^M}}} tokens. However, these are difficult to work with in general. We have now revised this behavior, such that end-of-line characters are converted to the \texttt{\obeyline} command when parsed by +v-type arguments. This change may require adjustments to the source of some documents, but the enhanced ability of users and programmers to exploit the +v-type argument means we believe it is necessary.

Declaring appropriate sub-encodings for TS1 symbol fonts
In 2020 we incorporated support for the TS1 symbol encoding directly into the kernel and in this way removed the need to load the textcomp package \cite{6} to make commands such as \texttt{\texteuro} available.

There is, however, a big problem with this TS1 symbol encoding: only very few fonts provide every glyph that is supposed to be part of TS1. This means that changing font families might result in certain symbols becoming unavailable. This can be a major disaster if, for example, the symbol \texttt{\texteuro} (€) or \texttt{\textohm} (Ω) no longer gets printed in your document, just because you altered the text font family.

To mitigate this problem, in 2020 we also introduced the declaration \texttt{\DeclareEncodingSubset}. This declaration is supposed to be used in font definition files for the TS1 encoding to specify which subset (we have defined 10 common ones) a specific font implements. If such a declaration is used then missing symbols are automatically taken from a fallback font.

While this is not perfect, it is the best you can do other than painstakingly checking that your document uses only glyphs that the font supports and, if necessary, switching to a different font or avoiding the missing symbols. See also the discussion in \cite{7}.

To jumpstart the process we also added declarations to the \text{T}e\text{X} kernel for most of the fonts found in \TeX{} Live at the time—with the assumption that such declarations would over time be superseded by declarations in the \texttt{.fd} files. Unfortunately, this hasn’t happened yet (or not often) and so many of the initial declarations went stale: several fonts got new glyphs added to them (so their sub-encoding should have been changed but didn’t); others (mainly due to license issues) changed the family name and thus our declarations became useless and the renamed fonts (now without a declaration) ended up in the default sub-encoding that offers only a few glyphs; yet others such as CharisSIL (which triggered
the GitHub issue) were simply not around at the time.

We have, therefore, again attempted to provide the (currently) correct declarations, but it is obvious that this is not a workable process. As we do not maintain the fonts we do not have the information that something has changed, and to regularly check the ever growing font support bundles is simply not possible. It is therefore very important that maintainers of font packages not only provide .fd files but also add such a declaration to every TS1...fd font definition file that they distribute.

To simplify this process, we now provide a simple \LaTeX file (checkencodingssubset.tex) for determining the correct (safe) sub-encoding. If run, it asks for a font family and then outputs its findings, for example, for \LaTeX Revived-TLF you will get:

```
Use sub-encoding 9
```

```
Testing font family \LaTeX Revived-TLF
(currently TS1-sub-encoding 9)

Some glyphs are missing from sub-encoding 8:
  \textcelsius (137) is missing
  \texttwosuperior (178) is missing
  \textthree (179) is missing
  \textonesuperior (185) is missing
  \texttwosuperior (178) is missing
  \textcelsius (137) is missing
  \textwon (142) is missing
  \texteuro (191) is missing
All glyphs between sub-encoding 6 and 7 exist
All glyphs between sub-encoding 5 and 6 exist
All glyphs between sub-encoding 4 and 5 exist
Some glyphs are missing from sub-encoding 3:
  \textmho (77) is missing
All glyphs between sub-encoding 2 and 3 exist
Some glyphs are missing from sub-encoding 1:
  \textiota (1) is missing
  \textmu (215) is missing
  \textmu (215) is missing
All glyphs between sub-encoding 0 and 1 exist
All glyphs in core exist

TS1 encoding subset for \LaTeX Revived-TLF (ok)
Use sub-encoding 9
```

This output is meant for human consumption, e.g., you see which glyphs are missing and why a certain sub-encoding is suggested, but it is not that hard to use it in a script and extract the suggested sub-encoding by grepping for the line starting with `Use sub-encoding`.

Of course, this check will only work if the missing glyphs are really missing; some fonts placed “tofu” into such slots and in this case it looks to \LaTeX as if the glyph is provided. For example, for the old Palatino fonts (family ppl) it would report

```
Use sub-encoding 0 (not 5)
```

thus it claims that all glyphs are provided, while in reality more than twenty are missing and sub-encoding 5, as declared in the kernel, is in fact correct.

```
(github issue 1257)
```

**Behavior when loading textcomp without options**

When incorporating the textcomp package into the \LaTeX kernel, in the February 2020 release [3], the default type of its package messages was changed from package info (\PackageTextcompInfo) to \LaTeX kernel info (\LaTeXInfo). But if textcomp was loaded without options, the message type got restored to package info. This restoration has now been canceled.

Note that loading textcomp with one of the options \error, \warn, or \info still changes the message type to an error, warning, or info message from the textcomp package.

```
(github issue 1333)
```

**Rollback improvements**

When requesting a rollback of the \LaTeX kernel and/or packages, several packages produced the error “Suspicious rollback date” because their rollback section contained only data about recent releases even if the package, such as array, was available since the first release of \LaTeX 2.ε in 1994. We now suppress this error and load the first release that is still part of the distribution (and hope for the best). This change was implemented for the packages amsmath, array, doc, graphics, longtable, multicol, showkeys, textcomp, and varioref.

```
(github issue 1333)
```

**Documentation improvements**

**Further updates to the guides**

We reported about the updated versions of usrguide and c1sguide in \LaTeX News 37 [6]. We have now revised fntguide as well to reflect the changes and macros added to the kernel over the last years of development. Note that the file name hasn’t changed and there is no fntguide-historic.

**Bug fixes**

**Fix inconsistent expansion of the package option list**

\LaTeX applies one-step expansion to the raw option list of packages and classes, so that constructions such as

\begin{verbatim}
\def\myoptions{opt1,opt2}
\usepackage[\myoptions]{foo}
\end{verbatim}

are supported. But if a package declares its options using the new key/value approach [5] and it gets loaded a second time, then its raw option list will not be expanded and so an error might be raised. This has now been corrected.

```
(github issue 1298)
```

\footnote{Little squares to indicate a missing symbol.}
Fix logic for first mark (page region)
In the new mark mechanism introduced in June 2022 [5] the result of \FirstMark on a two-column page was incorrect if the first column contained no marks. In that case it should have returned the first mark of the second column but didn’t. This has now been corrected.

Documents using \leftmark are not affected, because that command is still using the old mechanism for now. (github issue 1359)

Struts at the end of footnotes or p columns
To produce consistent spacing in footnotes and tabular p-cells LATEX adds a strut at the beginning and end of the content. This assumed, however, that the content of the footnote or tabular cell ended in vertical mode and so, until now, these struts were unconditionally added; as a result, if this content ended with vertical material then this strut started a new paragraph consisting of a single line with just the strut in it. This has finally been corrected and now the placement logic for the strut changes when vertical mode is detected.

(First seen in a bug report for footmisc in combination with bigfoot)

Fix a “missing \item” rollback error
If LATEX is rolled back to a date between 2023/06/01 (inclusive) and 2024/06/01 (exclusive), any list-based environment would raise an error

! LATEX Error: Something’s wrong--perhaps a missing \item\newline
This has now been corrected as a hotfix in patch level 2, by enhancing a 2023/06/01 version rollback code of new paragraph mechanism. (github issue 1386)

Changes to packages in the amsmath category
amsmath: Correct equation tag placement
If there is not enough space to place an equation tag on the same line as the equation then amsmath calculates a suitable offset and it places the tag above (or below) the equation. In the case of the gather environment this offset was not reset at the end, with the result that it also got applied to any following environment, resulting in incorrect spacing in certain situations. This has now been corrected. (github issue 1289)

Changes to packages in the tools category
array, longtable, tabularx: Support tagging
These three packages have been extended so they can now, on request, produce tagged tabular. This is done by adding a number of sockets (see [7]) that, by default, do nothing; but when tagged PDF is requested they get equipped with appropriate plugs.

In the previous LATEX release this was handled in \latex-lab, by patching the packages when tagging was requested.
array: No \unskip in math cells
Math cells in the standard array environment of the kernel are not subject to space removal at the right end of the cell, i.e., explicit spaces from \hspace or \!, etc. are honored (normal spaces are automatically ignored in math). In the array package all spaces got removed by calling \unskip unconditionally, regardless of the type of cell. This difference in behavior has now been removed by correcting the processing of math cells in array. (github issue 1323)
verbatim: \verb showed visible spaces
A recent change in the kernel was not reflected in the verbatim package, with the result that \verb showed visible spaces (\!) after the package was loaded. This has already been corrected in a hotfix for the November 2023 release. (github issue 1160)
verbatim: Support tabs in \verbatiminput*
Mimicking the November 2023 kernel update that allowed \verb* to mark tabs as spaces, the verbatim package has now been updated so that \verbatiminput* also marks tabs as spaces. (github issue 1245)
multicol: \columnbreak interferes with mark mechanism
The multicol package has to keep track of marks (from \markright or \markboth) as part of its output routine code and can’t rely on LATEX handling that automatically. It does so by artificially splitting page data with \vsplit to extract the mark data. With the introduction of \columnbreak that code failed sometimes, because it was not seeing any mark that followed such a forced column break.

This has now been corrected, but there is further work to do, because as of now multicol does not yet handle marks using the new mark mechanism—see the discussion at the beginning of the newsletter. (github issue 1130)
showkeys: Allow \newline in amsthm to work
Previously showkeys added an extra box layer which disabled the \newline of amsthm theorem styles. This extra box has now been avoided. (github issue 1123)
xr: Support links and properties
The xr package implements a system for eXternal References. The xr-hyper package (in the hyperref bundle) extended this to also support links to external documents. Using last year’s extension of the \label command, which unified the label syntax of LATEX and hyperref, it became possible to merge the two packages and thus make xr-hyper obsolete. With this change it is also possible to refer to properties that are stored in external documents using \RecordProperties. (github issue 1180)
Changes to files in the cyrillic category

Correct definition of \k

Ages ago, the encoding-specific definitions for various accent commands were changed to guard against altering some parameter values non-locally by mistake. For some reason the definition for \k in the Cyrillic encodings T2A, T2B, and T2C didn’t get this treatment. This oversight has now been corrected.  

(github issue 1148)

References


